5ª edición

Redes de computadoras

Un enfoque descendente



James F. Kurose
Keith W. Ross



ACCESO EN LÍNEA a *Redes de computadoras: un enfoque descendente,* 5.ª edición

Gracias por comprar una copia nueva de *Redes de computadoras: un enfoque descendente,* 5.ª edición. Su libro incluye seis meses de acceso prepagado al sitio web en inglés que complementa al libro. Esta suscripción gratuita le proporciona acceso completo a todas las áreas de apoyo al estudiante, incluyendo:

- Once prácticas de laboratorio con Wireshark.
- Nueve prácticas de programación.
- Applets en Java que ilustran diversos conceptos clave de las redes.
- Cuestionarios interactivos que le ayudarán a evaluar su grado de comprensión de los temas estudiados.
- Interesantes enlaces a otros recursos.

Por favor, lea a continuación las instrucciones de registro.
Para acceder por primera vez al sitio web de acompañamiento de *Redes de computadoras: un enfoque descendente,* 5.ª edición*:*
Tendrá que registrarse en línea mediante una computadora que disponga de conexión a Internet y que tenga instalado un explorador web. El proceso le llevará un par de minutos y sólo lo tendrá que hacer una vez.

1. Acceda a **http://www.aw.com/kurose_ross**
2. Haga clic en **Student Resources**.
3. Haga clic en el botón **Register**.
4. En la página de registro, introduzca su código de acceso de estudiante* que encontrará debajo de la franja de tinta removible. No escriba los guiones. Puede utilizar letras minúsculas o mayúsculas.
5. Siga las instrucciones indicadas en pantalla. Si necesita ayuda en cualquier momento durante el proceso de registro, simplemente haga clic en el icono **Need Help?**
6. Una vez que su nombre de usuario (***login name***) y su contraseña (***password***) estén confirmados, podrá comenzar a utilizar el sitio web de acompañamiento de *Redes de computadoras: un enfoque descendente*, 5.ª edición.

Cómo salir después de haberse registrado:
Sólo es necesario registrarse una vez en este sitio web.
Una vez que se haya registrado, podrá iniciar una sesión siempre que lo desee accediendo a **http://www.aw.com/kurose_ross** y proporcionando su nombre de usuario y su contraseña cuando se le soliciten.

*Importante: el código de acceso sólo se puede utilizar una vez. Esta suscripción es válida para un periodo de seis meses desde el momento de su activación y no es transferible. Si este código de acceso ya ha sido utilizado, es posible que ya no sea válido. Si se encuentra en este caso, puede adquirir una suscripción accediendo a **http://www.aw.com/kurose_ross** y siguiendo las instrucciones mostradas en pantalla.

REDES DE COMPUTADORAS

Un enfoque descendente

QUINTA EDICIÓN

REDES DE COMPUTADORAS

Un enfoque descendente

QUINTA EDICIÓN

James F. Kurose
University of Massachusetts, Amherst

Keith W. Ross
Politechnic Institute of NYU

REVISIÓN TÉCNICA

Carolina Mañoso Hierro
Profesora Titular de Universidad
Dpto. de Sistemas de Comunicación y Control
Escuela Técnica Superior de Ingeniería Informática
Universidad Nacional de Educación a Distancia

Ángel Pérez de Madrid y Pablo
Profesor Titular de Universidad
Dpto. de Sistemas de Comunicación y Control
Escuela Técnica Superior de Ingeniería Informática
Universidad Nacional de Educación a Distancia

REVISIÓN TÉCNICA PARA LATINOAMÉRICA

Luis Marrone
Ingeniero Electromecánico - Profesor Titular
Fac. de Informática, Universidad Nacional de
La Plata, Buenos Aires (Argentina)

Ingeniero Rubin Ayma Alejo Fedor
Director del Dpto. de Tecnología Informática
Fac. de Ingeniería, Universidad Argentina
de la Empresa, Buenos Aires (Argentina)

Paula Venosa
Licenciada en Informática - Profesora Adjunta
Fac. de Informática, Universidad Nacional de
La Plata, Buenos Aires (Argentina)

Ingeniero Carlos Alberto Binker
Coordinador de la especialidad de Redes de
Ingeniería en Informática
Universidad Nacional de la Matanza,
San Justo, Buenos Aires (Argentina)

Ingeniero Mario Groppo
Coordinador del Laboratorio de Redes del Departamento de Sistemas
Universidad Tecnológica Regional de Córdoba, Córdoba (Argentina)

Addison Wesley
es un sello editorial de



**Harlow, England • London • New York • Boston • San Francisco • Toronto • Sydney •
Singapore • Hong Kong • Tokyo • Seoul • Taipei • New Delhi • Cape Town
Madrid • Mexico City • Amsterdam • Munich • Paris • Milan**

REDES DE COMPUTADORAS: UN ENFOQUE DESCENDENTE
James F. Kurose, Keith W. Ross

PEARSON EDUCACIÓN, S. A. 2010
ISBN: 978-84-7829-119-9

Materia:, 004. Computadores

Formato: 195x250 mm Páginas: 844

Ribera del Loira, 28
28042 Madrid (España)

ISBN: 978-84-7829-119-9

Depósito Legal:

Traducción y Maquetación: Vuelapluma, S.L.U.

Equipo editorial:
Editor: Miguel Martín-Romo
Técnico Editorial: Esther Martín

Equipo de producción:
Director: José A. Clares
Técnico: Isabel Muñoz

Diseño de cubierta: Equipo de diseño de Pearson Educación, S. A.

Impreso por:

IMPRESO EN ESPAÑA - PRINTED IN SPAIN

Este libro ha sido impreso con papel y tintas ecológicos

A Julie y a nuestras tres
preciosidades: Chris, Charlie y Nina
JFK

A mi maravillosa esposa, Véronique,
y a nuestras tres hijas, Cécile, Claire y Katie
KWR

Acerca de los autores

Jim Kurose



Jim Kurose es profesor de Ciencias de la Computación en la Universidad de Massachusetts, Amherst.

Kurose ha recibido numerosos reconocimientos por sus actividades en el campo de la educación, entre los que se incluyen el premio a la excelencia en la labor pedagógica de la Universidad Tecnológica Nacional (ocho veces), la Universidad de Massachusetts y la asociación Northeast Association of Graduate Schools. Ha recibido la Medalla Taylor Booth del IEEE a la Educación y es bien conocido por haber liderado la iniciativa Commonwealth Information Technology de Massachusetts. Ha obtenido una beca de investigación GE Fellowship, un premio IBM al desarrollo del profesorado y una beca Lilly de enseñanza.

El Dr. Kurose ha sido editor jefe de las revistas *IEEE Transactions on Communications* e *IEEE/ACM Transactions on Networking*. Ha participado varios años en los comités de programa de las conferencias *IEEE Infocom, ACM SIGCOMM, ACM Internet Measurement Conference* y *ACM SIGMETRICS* y ha sido también Co-director Técnico de Programa para dichas conferencias. Es miembro del IEEE de la ACM. Entre sus intereses de investigación se incluyen las arquitecturas y protocolos de red, las medidas de red, las redes de sensores, la comunicación multimedia y el modelado y la evaluación de rendimiento. Es doctor en Ciencias de la Computación por la Universidad de Columbia.

Keith Ross



Keith Ross es Jefe del Departamento de Ciencias de la Computación en el Instituto Politécnico de la Universidad de Nueva York (en Brooklyn) y cuenta con la distinción pedagógica Leonard J. Shustek de dicho departamento. Entre 1985 y 1998 fue profesor del Departamento de Ingeniería de Sistemas en la Universidad de Pennsylvania. De 1998 a 2003 trabajó como profesor en el Departamento de Comunicaciones Multimedia del Instituto Eurecom en Francia. Keith Ross fue también el fundador principal y primer consejero delegado de Wimba, que desarrolla tecnologías de Voz sobre IP y transmisión de flujos multimedia para el mercado de la enseñanza electrónica.

Entre sus intereses de investigación se encuentran las redes P2P, las medidas para Internet, la tecnología de flujos de vídeo, las cachés web, las redes de distribución de contenido, la seguridad de red, Voz sobre IP y el modelado estocástico. Es miembro del IEEE y actualmente trabaja como editor asociado para la revista *IEEE/ACM Transactions on Networking*. Ha actuado como asesor para la Comisión Federal de Comercio sobre compartición de archivos P2P y ha formado también parte de los comités de programa de las conferencias *IEEE Infocom, ACM SIGCOMM, ACM CoNext, IPTPS, ACM Multimedia, ACM Internet Measurement Conference* y *ACM SIGMETRICS*. Es doctor en Ingeniería de computación, información y control por la Universidad de Michigan.

Prefacio

Bienvenido a la quinta edición de *Redes de computadoras: un enfoque descendente*. Desde la publicación de la primera edición hace nueve años, nuestro libro ha sido adoptado por centenares de universidades, traducido a más de una docena de idiomas y utilizado por más cien mil estudiantes y usuarios de todo el mundo. Hemos tenido noticias de muchos de estos lectores y estamos abrumados por su positiva respuesta.

Novedades en la quinta edición

Creemos que una razón importante de este éxito ha sido que nuestro libro continúa ofreciendo un enfoque novedoso y oportuno para la formación en el campo de las redes de computadoras. Hemos realizado cambios en esta quinta edición, pero también hemos dejado todo aquello que creemos que constituyen (y en lo que coinciden los estudiantes y los profesores que han utilizado nuestro libro) los aspectos más importantes del libro: su enfoque descendente, el hecho de que está centrado en Internet y en un tratamiento moderno de las redes de computadoras, su atención tanto a los principios como a la práctica y su estilo y enfoque pedagógico accesibles en lo que respecta al aprendizaje de las redes de computadoras.

De todos modos, se ha realizado una serie de cambios importantes en la quinta edición. En el Capítulo 1, hemos actualizado nuestra introducción al tema de las redes y también hemos actualizado y ampliado el tratamiento de las redes de acceso (en particular, el uso de redes de cable, DSL y redes de fibra como redes de acceso a la Internet pública). En el Capítulo 2, hemos eliminado el material sobre búsquedas entre pares que había quedado un tanto obsoleto, con el fin de dejar espacio para una nueva sección sobre tablas hash distribuidas. Como siempre, cuando se ha eliminado material del libro, éste sigue estando disponible en el sitio web de acompañamiento (véase más adelante). La presentación de los sistemas de control de congestión de TCP en el Capítulo 3 está basada ahora en una representación gráfica (máquina de estados finitos) de TCP, lo que añade una mejor estructura y claridad a la exposición. El Capítulo 5 se ha ampliado significativamente con nuevas secciones dedicadas a las redes de área local virtuales (VLAN) y al análisis detallado del proceso que sigue una solicitud web. En esta última sección se realiza un seguimiento de toda la actividad de red y los protocolos implicados en el proceso de satisfacer la aparentemente simple solicitud de extracción y visualización de una página web almacenada en un servidor remoto, lo que ayuda a ilustrar y sintetizar buena parte del material cubierto en los primeros cinco capítulos. En el Capítulo 6, hemos eliminado parte de la "sopa de letras" de estándares y protocolos de telefonía celular y hemos añadido una nueva sección sobre la arquitectura de las redes celulares y cómo interoperan las redes celulares e Internet para proporcionar servicios de Internet a dispositivos móviles, tales como un teléfono Blackberry o un iPhone. El tratamiento de la seguridad en las redes en el Capítulo 8 ha sufrido una importante revisión. Se ha revisado el material dedicado a la autenticación de punto terminal, al encadenamiento de sistemas de cifrado de bloque y a la criptografía de clave pública, y el material sobre IPsec ha sido reescrito y ampliado con el fin de incluir el tema de la redes privadas virtuales (VPN). A lo largo del libro, hemos incluido nuevos ejemplos avanzados y referencias actualizadas. En el mate-

rial incluido al final de cada capítulo, hemos añadido nuevos problemas (como nos habían solicitado muchos profesores), hemos portado las prácticas de laboratorio de Ethereal a Wireshark, hemos añadido nuevas prácticas de laboratorio con Wireshark y también una nueva práctica de laboratorio sobre IPsec.

Audiencia

Este libro de texto es para un primer curso sobre redes de computadoras. Se puede utilizar tanto en departamentos de informática como de ingeniería eléctrica. En relación con los lenguajes de programación, se supone que el estudiante sólo tiene experiencia con C, C++ o Java (e incluso esa suposición sólo se hace en algunos pocos lugares). Aunque este libro es más preciso y analítico que muchos otros textos introductorios a las redes de computadoras, rara vez utiliza conceptos matemáticos que no se hayan aprendido en el bachillerato. Hemos hecho un esfuerzo deliberado por evitar el uso de cualquier concepto de cálculo avanzado, probabilidad o procesos estocásticos (aunque hemos incluido algunos problemas para los estudiantes que dominen estos conceptos). El libro es apropiado por tanto para cursos de primer ciclo y para el primer año de los cursos de segundo ciclo. También puede ser adecuado para los trabajadores del sector de las telecomunicaciones.

¿Qué hace especial a este libro de texto?

El tema de las redes de computadoras es enormemente complejo, implicando muchos conceptos, protocolos y tecnologías que están entrelazados de una manera intrincada. Para abarcar este alcance y complejidad, muchos textos de redes se han organizado a menudo sobre las "capas" de una arquitectura de red. Con una organización en capas, los estudiantes pueden ver a través de la complejidad de las redes de computadoras y aprender los distintos conceptos y protocolos de una parte de la arquitectura a la vez que ven el gran esquema de cómo todas las partes se ajustan entre sí. Desde una perspectiva pedagógica, nuestra experiencia personal ha sido que dicho enfoque en capas es efectivamente muy deseable. Sin embargo, hemos comprobado que el enfoque tradicional de enseñanza ascendente, es decir, desde la capa física a la capa de aplicación, no es el mejor enfoque para un curso moderno sobre redes de computadoras.

Un enfoque descendente

Nuestro libro rompió moldes hace diez años al abordar el tema de las redes de arriba hacia abajo; es decir, comenzando con la capa de aplicación y descendiendo desde allí hacia la capa física. Los comentarios recibidos tanto de profesores como de estudiantes nos confirman que el enfoque descendente tiene muchas ventajas y resulta pedagógicamente adecuado. En primer lugar, hace énfasis en la capa de aplicación (un "área de elevado crecimiento" en las redes). De hecho, muchas revoluciones recientes en las redes de computadoras, incluyendo la Web, la compartición de archivos P2P y los flujos de audio y vídeo, han tenido lugar en la capa de aplicación. Un énfasis inicial en la capa de aplicación difiere de los métodos considerados en la mayor parte de otros textos, que incluyen sólo una pequeña cantidad de material sobre las aplicaciones de red, sus requisitos, paradigmas (por ejemplo, cliente-servidor y P2P) e interfaces de programación de aplicaciones. En

segundo lugar, nuestra experiencia como profesores (y de muchos profesores que han utilizado este texto) ha sido que el enseñar las aplicaciones de redes al principio de curso es una potente herramienta de motivación. Los estudiantes se emocionan al aprender cómo funcionan las aplicaciones de red, aplicaciones tales como el correo electrónico o la Web, que la mayoría de los estudiantes utilizan diariamente. Una vez que los estudiantes comprenden las aplicaciones, pueden entonces entender los servicios de red necesarios para proporcionarles ese soporte. El estudiante puede entonces, a su vez, examinar las distintas formas en que tales servicios pueden ser suministrados por, e implementados en, las capas inferiores. Por tanto, tratar las aplicaciones inicialmente proporciona motivación para abordar el resto del texto.

En tercer lugar, un enfoque descendente permite a los profesores introducir el desarrollo de aplicaciones de red en una etapa temprana. Los estudiantes no sólo ven cómo funciona una serie de populares aplicaciones y protocolos, sino que también aprenden lo fácil que es crear sus propias aplicaciones de red y protocolos del nivel de aplicación. Con el enfoque descendente, los estudiantes descubren pronto las nociones acerca de las interfaces de programación de aplicaciones (API), los modelos de servicio y los protocolos (conceptos importantes que vuelven a serlo en todas las capas subsiguientes). Al proporcionar ejemplos de programación de sockets en Java, destacamos las ideas centrales sin confundir a los estudiantes con fragmentos complejos de código. Los estudiantes de primer ciclo de Ingeniería Eléctrica y Ciencias de la Computación no deben tener dificultades para comprender el código Java.

Un enfoque Internet

Aunque en la cuarta edición del libro eliminamos del título la frase "Caracterización de Internet", ¡no quiere decir que hayamos eliminado el enfoque de Internet! ¡Nada más lejos de la realidad! En lugar de ello, y dado que Internet se ha vuelto tan ubicua, pensamos que cualquier libro de texto sobre redes debe centrarse significativamente en Internet, por lo que la frase del título era hasta cierto punto innecesaria. En el texto, continuamos utilizando la arquitectura y los protocolos de Internet como vehículo principal para estudiar los conceptos fundamentales de las redes de computadoras. Por supuesto, también incluimos conceptos y protocolos de las arquitecturas de red, pero el foco está centrado claramente en Internet, un hecho que se ha reflejado en la organización del libro, que está centrada alrededor de la arquitectura de cinco capas de Internet: las capas de aplicación, transporte, red, enlace y física.

Otro de los beneficios de centrar la atención en Internet es que la mayoría de los estudiantes de Ciencias de la Computación y de Ingeniería Eléctrica están deseosos de aprender cosas sobre Internet y sus protocolos. Saben que Internet ha sido una tecnología revolucionaria y conflictiva que está cambiando profundamente nuestro mundo. Dada la enorme relevancia de Internet, los estudiantes sienten una natural curiosidad por ver lo que hay "entre bastidores". Por tanto, es fácil para un profesor conseguir que los estudiantes se interesen por los principios básicos cuando se utiliza Internet como foco guía.

Cómo enseñar los principios de las redes

Dos de las características únicas del libro (su enfoque descendente y su foco puesto en Internet) se han utilizado en el título en sucesivas ediciones. Si hubiéramos podido introducir una *tercera* frase en el subtítulo, ésta habría contenido la palabra *principios.* El campo de las

redes está ahora lo suficientemente maduro como para que se puedan identificar una serie de temas fundamentalmente importantes. Por ejemplo, en la capa de transporte, los temas fundamentales son la comunicación fiable sobre una capa de red no fiable, el establecimiento y el cierre de la conexión y el proceso de acuerdo, el control de congestión y flujo, y la multiplexación. Dos temas enormemente importantes de la capa de red son la determinación de "buenas" rutas entres dos routers y la interconexión de un número grande redes heterogéneas. En la capa de enlace de datos, un problema fundamental es la compartición de un canal de acceso múltiple. En el campo de la seguridad de red, las técnicas que permiten proporcionar confidencialidad, autenticación e integridad de los mensajes están basadas en los principios de la criptografía. Este texto identifica los temas fundamentales acerca de las redes y estudia los métodos para abordarlos. El estudiante que aprenda estos principios adquirirá una serie de conocimientos con una larga "vida útil": mucho después de que los estándares y protocolos de red actuales hayan quedado obsoletos, los principios en que se basan continuarán teniendo importancia y siendo relevantes. Pensamos que la combinación del uso de Internet para atisbar las posibilidades y el énfasis posterior en los temas fundamentales y las soluciones permitirán al estudiante comprender rápidamente cualquier tecnología de redes.

El sitio web

Al adquirir este libro de texto, el lector podrá acceder durante seis meses al sitio web de acompañamiento en la dirección http://www.aw.com/kurose-ross. Este sitio incluye:

- *Material de aprendizaje interactivo.* El sitio web contiene varios applets Java interactivos, que ilustran muchos de los conceptos clave de redes. El sitio también ofrece preguntas de test interactivas que permiten a los estudiantes comprobar su conocimiento básico del tema en cuestión. Los profesores pueden integrar estas características interactivas en sus clases o utilizarlas como mini prácticas de laboratorio.
- *Material técnico adicional.* A medida que hemos ido añadiendo nuevo material en cada edición del libro, hemos tenido que eliminar parte del tratamiento de alguno de los temas existentes, para que el libro tuviera una longitud aceptable. Por ejemplo, para hacer sitio para el nuevo material incluido en esta edición, hemos eliminado diversos materiales sobre redes ATM y búsquedas P2P. Pero el material incluido en las ediciones anteriores del libro sigue de interés y puede encontrarse en el sitio web de acompañamiento.
- *Tareas de programación.* El sitio web también proporciona una serie de tareas de programación detalladas, entre las que se incluyen la construcción de un servidor web multihebra, la construcción de un cliente de correo electrónico con una interfaz gráfica de usuario (GUI), la programación de los lados emisor y receptor de un protocolo de transporte de datos fiable, la programación de un algoritmo de enrutamiento distribuido y otros.
- *Prácticas de laboratorio con Wireshark.* La comprensión de los protocolos de red puede verse ampliada significativamente viendo esos protocolos en acción. El sitio web proporciona numerosas prácticas con Wireshark que permiten a los estudiantes observar la secuencia de mensajes intercambiados entre dos entidades de protocolo. El sitio web incluye prácticas de laboratorio con Wireshark independientes sobre HTTP, DNS, TCP, UDP, IP, ICMP, Ethernet, ARP, WiFi, SSL, y sobre cómo realizar una traza de todos los protocolos implicados en satisfacer una solicitud de extracción de una página web. Continuaremos añadiendo nuevas prácticas de laboratorio en el futuro.

Características pedagógicas

Cada uno de nosotros lleva enseñando redes de computadoras durante más de 20 años. Proporcionamos a este texto la experiencia combinada de enseñar durate 45 años a muchos miles de estudiantes. Hemos sido también investigadores activos en redes de computadoras durante este tiempo. (De hecho, Jim y Keith se conocieron cuando eran estudiantes de máster en un curso sobre redes de computadoras impartido por Mischa Schwartz en 1979 en la Universidad de Columbia.) Pensamos que todo esto nos proporciona una buena perspectiva de cuál ha sido la evolución de las redes y hacia dónde es probable que vayan en el futuro. Sin embargo, hemos resistido a la tentación de dirigir el material de este libro hacia las preferencias de nuestros proyectos de investigación. Consideramos que el lector puede visitar nuestros sitios web personales si está interesado en nuestras investigaciones. Por tanto, este libro se ocupa de las redes de computadoras modernas (de los protocolos y tecnologías contemporáneos, así como de los principios subyacentes a dichos protocolos y tecnologías). También creemos que aprender (y enseñar) redes puede ser divertido. Esperamos que el sentido del humor, el uso de analogías y los ejemplos del mundo real contenidos en el libro hagan el material más divertido.

Notas sobre historia, práctica y seguridad

El campo de las redes de computadoras tiene una rica y fascinante historia. Hemos hecho un esfuerzo especial para contar la historia de las redes de computadoras. Hemos incluido una sección histórica especial en el Capítulo 1 y una docena de notas históricas distribuidas por el resto de los capítulos. En estas historias hemos hablado de la invención de la conmutación de paquetes, de la evolución de Internet, del nacimiento de los principales gigantes de las redes como Cisco y 3Com, y de otros muchos hechos importantes. Los estudiantes se sentirán estimulados por estas historias. Hemos incluido asimismo notas especiales que destacan los principios importantes en las redes de computadoras. Estas notas ayudarán a los estudiantes a apreciar algunos de los conceptos fundamentales que se aplican en las redes modernas. El tratamiento de los temas de seguridad en las redes se ha ampliado, incluyendo por ejemplo las notas especiales "Seguridad" en cada uno de los capítulos fundamentales del libro.

Entrevistas

También hemos incluido otras característica original que nuestros lectores nos dicen que les resulta particularmente interesante e inspiradora: las entrevistas con innovadores de renombre en el campo de las redes. Hemos entrevistado a Len Kleinrock, Bram Cohen, Sally Floyd, Vint Cerf, Simon Lam, Charlie Perkins, Henning Schulzrinne, Steven Bellovin y Jeff Case.

Suplementos para los profesores

Proporcionamos un paquete de suplementos completo para ayudar a los profesores a la hora de impartir este curso. Se puede acceder a este material en el Centro de recursos de formación de Addison-Wesley (http://www.pearsonhighered.com/irc). Visite este centro o envíe un correo electrónico a computing@aw.com para obtener información sobre cómo acceder a los suplementos para los profesores.

- *Diapositivas PowerPoint*®. Proporcionamos diapositivas para los nueve capítulos. Las diapositivas para esta quinta edición se han actualizado de forma significativa. Estas presentaciones en diapositivas cubren cada capítulo en detalle. Se utilizan gráficos y animaciones (en lugar de emplear únicamente monótonas listas de viñetas) que hacen que las diapositivas sean interesantes y visualmente atractivas. Proporcionamos las diapositivas originales de PowerPoint de modo que pueda personalizarlas con el fin de adaptarlas a sus necesidades a la hora de impartir el curso. Algunas de estas diapositivas han sido aportadas por otros profesores que han enseñado con nuestro libro.
- *Soluciones de los problemas.* Proporcionamos un manual de soluciones para los problemas incluidos en el texto, las tareas de programación y las prácticas de laboratorio de Wireshark. Como hemos dicho anteriormente, hemos añadido muchos problemas nuevos en los primeros cinco capítulos del libro.

Dependencias entre capítulos

El primer capítulo de este texto presenta una panorámica general autocontenida de las redes de computadoras, introduciéndose muchos conceptos clave y terminología; este capítulo define el escenario para el resto del libro. Todos los capítulos restantes dependen de este primer capítulo. Recomendamos a los profesores que, después de completar el Capítulo 1, aborden los Capítulos 2 a 5 en orden, siguiendo nuestra filosofía del enfoque descendente. Cada uno de estos cinco capítulos se apoya en el material de los capítulos anteriores. Una vez completados los primeros cinco capítulos, el profesor tiene bastante flexibilidad. No existen interdependencias entre los cuatro últimos capítulos, por lo que se pueden abordar en cualquier orden. Sabemos que algunos profesores después de explicar el capítulo de introducción, saltan al Capítulo 5 y van hacia atrás (enfoque ascendente) e incluso otro profesor que comienza por la mitad (Capítulo 4) y luego va progresando en ambas direcciones. Sin embargo, cada uno de los últimos cuatro capítulos depende del material de los primeros cinco. Muchos profesores explican los primeros cinco capítulos y luego uno de los cuatro últimos como “postre”.

Una última nota: agradecemos cualquier comentario

Animamos a los estudiantes y profesores a enviarnos por correo electrónico cualquier comentario que tengan sobre nuestro libro. Ha sido maravilloso para nosotros escuchar opiniones de profesores y estudiantes de todo el mundo sobre las cuatro primeras ediciones. Muchas de esas sugerencias se han incorporado en ediciones posteriores del libro. También animamos a los profesores a enviarnos nuevos problemas (y sus soluciones) que complementen los problemas actualmente incluidos, los cuales añadiremos únicamente en la parte de acceso exclusivo para profesores del sitio web. Animamos también a los profesores y estudiantes a crear nuevos applets Java que ilustren los conceptos y protocolos de este libro. Si tiene un applet que piensa que podría ser adecuado para este texto, por favor, envienoslo. Si el applet (incluyendo la notación y terminología) es apropiado, estaremos encantados de incluirlo en el sitio web del libro, junto con la apropiada referencia a los autores del mismo.

Siéntase libre de enviarnos URL interesantes, indíquenos los errores tipográficos, disienta de nuestras afirmaciones y díganos lo que cree que funciona y lo que no. Díga-

nos qué es los que piensa que debería o no debería ser incluido en la siguiente edición. Envíenos su correo electrónico a kurose@cs.umass.edu y ross@poly.edu.

Agradecimientos

Desde que comenzamos a escribir este libro en 1996, muchas personas nos han proporcionado una ayuda inestimable y nos ha influido dando forma a nuestras ideas sobre cómo organizar e impartir un curso de redes. MUCHAS GRACIAS a todos los que nos han ayudado desde el primer borrador del libro hasta la quinta edición. Estamos también *muy* agradecidos a los muchos cientos de lectores de todo el mundo (estudiantes, profesores, usuarios) que nos han enviado ideas y comentarios sobre las ediciones anteriores del libro y sugerencias sobre las futuras ediciones del mismo. Gracias especiales a:

Al Aho (Universidad de Columbia)
Hisham Al-Mubaid (Universidad de Houston-Clear Lake)
Pratima Akkunoor (Universidad del Estado de Arizona)
Paul Amer (Universidad de Delaware)
Shamiul Azom (Universidad del Estado de Arizona)
Lichun Bao (Universidad de California en Irvine)
Paul Barford (Universidad de Wisconsin)
Bobby Bhattacharjee (Universidad de Maryland)
Steven Bellovin (Universidad de Columbia)
Pravin Bhagwat (Wibhu)
Supratik Bhattacharyya (anteriormente en Sprint)
Ernst Biersack (Instituto Eurécom)
Shahid Bokhari (Universidad de Ingeniería y Tecnología, Lahore)
Jean Bolot (Sprint)
Daniel Brushteyn (antigua Universidad de Pensilvania)
Ken Calvert (Universidad de Kentucky)
Evandro Cantu (Universidad Federal de Santa Catarina)
Jeff Case (SNMP Research International)
Jeff Chaltas (Sprint)
Vinton Cerf (Google)
Byung Kyu Choi (Universidad Tenológica de Michigan)
Bram Cohen (BitTorrent, Inc.)
Constantine Coutras (Pace University)
John Daigle (Universidad de Mississippi)
Edmundo A. de Souza e Silva (Universidad Federal de Río de Janiero)
Philippe Decuetos (Instituto Eurécom)
Christophe Diot (Thomson Research)
Prithula Dhunghel (Instituto Politécnico de NYU)
Michalis Faloutsos (Universidad de California en Riverside)
Wu-chi Feng (Oregon Graduate Institute)
Sally Floyd (ICIR, Universidad de California en Berkeley)
Paul Francis (Instituto Max Planck)
Lixin Gao (Universidad de Massachusetts)

JJ Garcia-Luna-Aceves (Universidad de California en Santa Cruz)
Mario Gerla (Universidad de California de Los Angeles)
David Goodman (Universidad Politécnica)
Tim Griffin (Universidad de Cambridge)
Max Hailperin (Gustavus Adolphus College)
Bruce Harvey (Florida A&M University, Florida State University)
Carl Hauser (Universidad del Estado de Washington)
Rachelle Heller (George Washington University)
Phillipp Hoschka (INRIA/W3C)
Wen Hsin (Park University)
Albert Huang (former University of Pennsylvania student)
Esther A. Hughes (Virginia Commonwealth University)
Jobin James (Universidad de California en Riverside)
Sugih Jamin (Universidad de Michigan)
Shivkumar Kalyanaraman (Rensselaer Polytechnic Institute)
Jussi Kangasharju (Universidad de Darmstadt)
Sneha Kasera (Universidad de Utah)
Hyojin Kim (former Universidad de Pennsylvania student)
Leonard Kleinrock (Universidad de California en Los Angeles)
David Kotz (Dartmouth College)
Beshan Kulapala (Universidad del Estado de Arizona)
Rakesh Kumar (Bloomberg)
Miguel A. Labrador (Universidad de South Florida)
Simon Lam (Universidad de Texas)
Steve Lai (Universidad del Estado de Ohio)
Tom LaPorta (Universidad de Penn State)
Tim-Berners Lee (World Wide Web Consortium)
Lee Leitner (Universidad de Drexel)
Brian Levine (Universidad de Massachusetts)
William Liang (antigua Universidad de Pensilvania)
Willis Marti (Universidad de Texas A&M)
Nick McKeown (Universidad de Stanford)
Josh McKinzie (Universidad de Park)
Deep Medhi (Universidad de Missouri, Kansas City)
Bob Metcalfe (International Data Group)
Sue Moon (KAIST)
Erich Nahum (IBM Research)
Christos Papadopoulos (Universidad del Estado de Colorado)
Craig Partridge (Tecnologías BBN)
Radia Perlman (Sun Microsystems)
Jitendra Padhye (Microsoft Research)
Vern Paxson (Universidad de California en Berkeley)
Kevin Phillips (Sprint)
George Polyzos (Universidad de Atenas of Economics and Business)
Sriram Rajagopalan (Universidad del Estado de Arizona)
Ramachandran Ramjee (Microsoft Research)
Ken Reek (Instituto de Tecnología de Rochester)

Martin Reisslein (Universidad del Estado de Arizona)
Jennifer Rexford (Universidad de Princeton)
Leon Reznik (Instituto de Tecnología Rochester)
Sumit Roy (Universidad de Washington)
Avi Rubin (Universidad de Johns Hopkins)
Dan Rubenstein (Universidad de Columbia)
Douglas Salane (John Jay College)
Despina Saparilla (Cisco Systems)
Henning Schulzrinne (Universidad de Columbia)
Mischa Schwartz (Universidad de Columbia)
Harish Sethu (Universidad de Drexel)
K. Sam Shanmugan (Universidad de Kansas)
Prashant Shenoy (Universidad de Massachusetts)
Clay Shields (Universidad de Georgetown)
Subin Shrestra (Universidad de Pensilvania)
Mihail L. Sichitiu (Universidad de NC State)
Peter Steenkiste (Universidad de Carnegie Mellon)
Tatsuya Suda (Universidad de California en Irvine)
Kin Sun Tam (Universidad del Estado de Nueva York en Albany)
Don Towsley (Universidad de Massachusetts)
David Turner (Universidad del Estado de California, San Bernardino)
Nitin Vaidya (Universidad de Illinois)
Michele Weigle (Universidad de Clemson)
David Wetherall (Universidad de Washington)
Ira Winston (Universidad de Pensilvania)
Di Wu (Instituto Politécnico de NYU)
Raj Yavatkar (Intel)
Yechiam Yemini (Universidad de Columbia)
Ming Yu (Universidad del Estado de Nueva York en Binghamton)
Ellen Zegura (Instituto de Tecnología de Georgia)
Honggang Zhang (Universidad de Suffolk)
Hui Zhang (Universidad de Carnegie Mellon)
Lixia Zhang (Universidad de California en Los Angeles)
Shuchun Zhang (former Universidad de Pennsylvania student)
Xiaodong Zhang (Universidad del Estado de Ohio)
ZhiLi Zhang (Universidad de Minnesota)
Phil Zimmermann (consultor independiente)
Cliff C. Zou (Universidad de Central Florida)

Queremos dar las gracias al profesor Honggang Zhang de la Universidad de Suffolk por trabajar con nosotros durante la revisión y mejora de los conjuntos de problemas de esta edición. También deseamos dar las gracias al equipo completo de Addison-Wesley, y en particular a Michael Hirsch, Marilyn Lloyd y Stephanie Sellinger, que han hecho un trabajo absolutamente excelente en esta quinta edición (y que han sabido llevar a dos autores que parecen congénitamente incapaces de cumplir con los plazos). Gracias también a nuestros artistas, Janet Theurer y Patrice Rossi Calkin, por su trabajo en las bonitas figuras del libro, así como a Nesbitt Graphics, Harry Druding y Rose Kernan por el maravilloso trabajo de pro-

ducción de esta edición. Por último, una gratitud muy especial a Michael Hirsch, nuestro editor en Addison-Wesley, y Susan Hartman, nuestra anterior editora de Addison-Wesley. Este libro no sería lo que es (e incluso puede que nunca hubiera llegado a ser) sin su apropiada gestión del proyecto, su apoyo constante, su paciencia casi infinita, su buen humor y su perseverancia.

Contenido



CAPÍTULO 1

Redes de computadoras e Internet

Hoy día, Internet es casi indiscutiblemente el sistema de ingeniería más grande creado por la mano del hombre, con cientos de millones de computadoras conectadas, enlaces de comunicaciones y switches; cientos de millones de usuarios que se conectan de forma intermitente a través de sus teléfonos móviles y sus PDA; y dispositivos tales como sensores, cámaras web, consolas de juegos, marcos de fotografías e incluso lavadoras que se conectan a Internet. Dado que Internet es una red tan enorme e incluye tantos componentes distintos y tiene tantos usos, ¿es posible tener la esperanza de comprender cómo funciona y, más concretamente, cómo funcionan las redes de computadoras? ¿Existen unos principios y una estructura básicos que puedan proporcionar una base para comprender un sistema tan asombrosamente complejo y grande? Y, en caso afirmativo, ¿es posible que pueda resultar tan interesante y divertido como para dedicarse a estudiar las redes de computadoras? Afortunadamente, la respuesta a todas estas preguntas es un rotundo SÍ. Ciertamente, el objetivo de este libro es el de iniciar al lector en el dinámico campo de las redes de computadoras, proporcionándole los principios y los conocimientos prácticos que necesitará para entender no sólo las redes actuales, sino también las del futuro.

En el primer capítulo se hace una amplia introducción al mundo de las redes de computadoras y de Internet. Nuestro objetivo es proporcionar una visión general y establecer el contexto para el resto del libro, con el fin de poder ver el bosque a través de los árboles. En este capítulo de introducción se abordan muchos de los fundamentos, así como muchos de los componentes que forman una red de computadoras, siempre sin perder de vista la panorámica general.

Vamos a estructurar esta introducción a las redes de computadoras de la siguiente forma: después de exponer algunos conceptos y términos fundamentales, examinaremos los componentes esenciales que forman una red de computadoras. Comenzaremos por la frontera de la red y echaremos un vistazo a los sistemas terminales y aplicaciones que se ejecu-

tan en la red. A continuación, exploraremos el núcleo de una red de computadoras, examinando los enlaces y los switches que transportan los datos, así como las redes de acceso y los medios físicos que conectan los sistemas terminales con el núcleo de la red. Aprenderemos que Internet es una red de redes y cómo estas redes se conectan entre sí.

Una vez completada la introducción sobre la frontera y el núcleo de una red de computadoras, en la segunda mitad del capítulo adoptaremos un punto de vista más amplio y abstracto. Examinaremos los retardos, las pérdidas y la tasa de transferencia de datos en una red de computadoras y proporcionaremos modelos cuantitativos simples para los retardos y tasas de transferencia de terminal a terminal: modelos que tienen en cuenta los retardos de transmisión, de propagación y de cola. A continuación, presentaremos algunos de los principios básicos sobre las arquitecturas de red, en concreto, las capas de protocolos y los modelos de servicios. También veremos que las redes son vulnerables a muchos tipos distintos de ataques; revisaremos algunos de estos ataques y veremos cómo es posible conseguir que las redes sean más seguras. Por último, concluiremos el capítulo con una breve historia de las redes de comunicaciones.

1.1 ¿Qué es Internet?

En este libro, vamos a emplear la red pública Internet, una red de computadoras específica, como nuestro principal vehículo para explicar las redes de computadoras y sus protocolos. Pero, ¿qué es Internet? Hay dos formas de responder a esta pregunta. La primera de ellas es describiendo las tuercas y tornillos que forman la red; es decir, los componentes hardware y software básicos que forman Internet. La segunda es describiéndola en términos de la infraestructura de red que proporciona servicios a aplicaciones distribuidas. Comenzaremos por la descripción de los componentes esenciales, utilizando la Figura 1.1 para ilustrar la exposición.

1.1.1 Descripción de los componentes esenciales

Internet es una red de computadoras que interconecta cientos de millones de dispositivos informáticos a lo largo de todo el mundo. No hace demasiado tiempo, estos dispositivos eran fundamentalmente computadoras PC de escritorio tradicionales, estaciones de trabajo Linux y los llamados servidores que almacenaban y transmitían información tal como páginas web y mensajes de correo electrónico. Sin embargo, cada vez más sistemas terminales no tradicionales como televisiones, computadoras portátiles, consolas de juegos, teléfonos móviles, cámaras web, sistemas de detección medioambientales y de automóviles y dispositivos de seguridad y electrodomésticos están conectados a Internet. Por tanto, el término *red de computadoras* está comenzando a quedar algo desactualizado a causa de la gran cantidad de dispositivos no tradicionales que están conectados a Internet. En la jerga de Internet, todos estos dispositivos se denominan **hosts** o **sistemas terminales**. En julio de 2008, había casi 600 millones de sistemas terminales conectados a Internet [ISC 2009], sin contar los teléfonos móviles, las computadoras portátiles y otros dispositivos que se conectan de forma intermitente a Internet.

Los sistemas terminales se conectan entre sí mediante una red de **enlaces de comunicaciones** y dispositivos de **conmutación de paquetes**. En la Sección 1.2 veremos que existen muchos tipos de enlaces de comunicaciones, los cuales están compuestos por diferentes

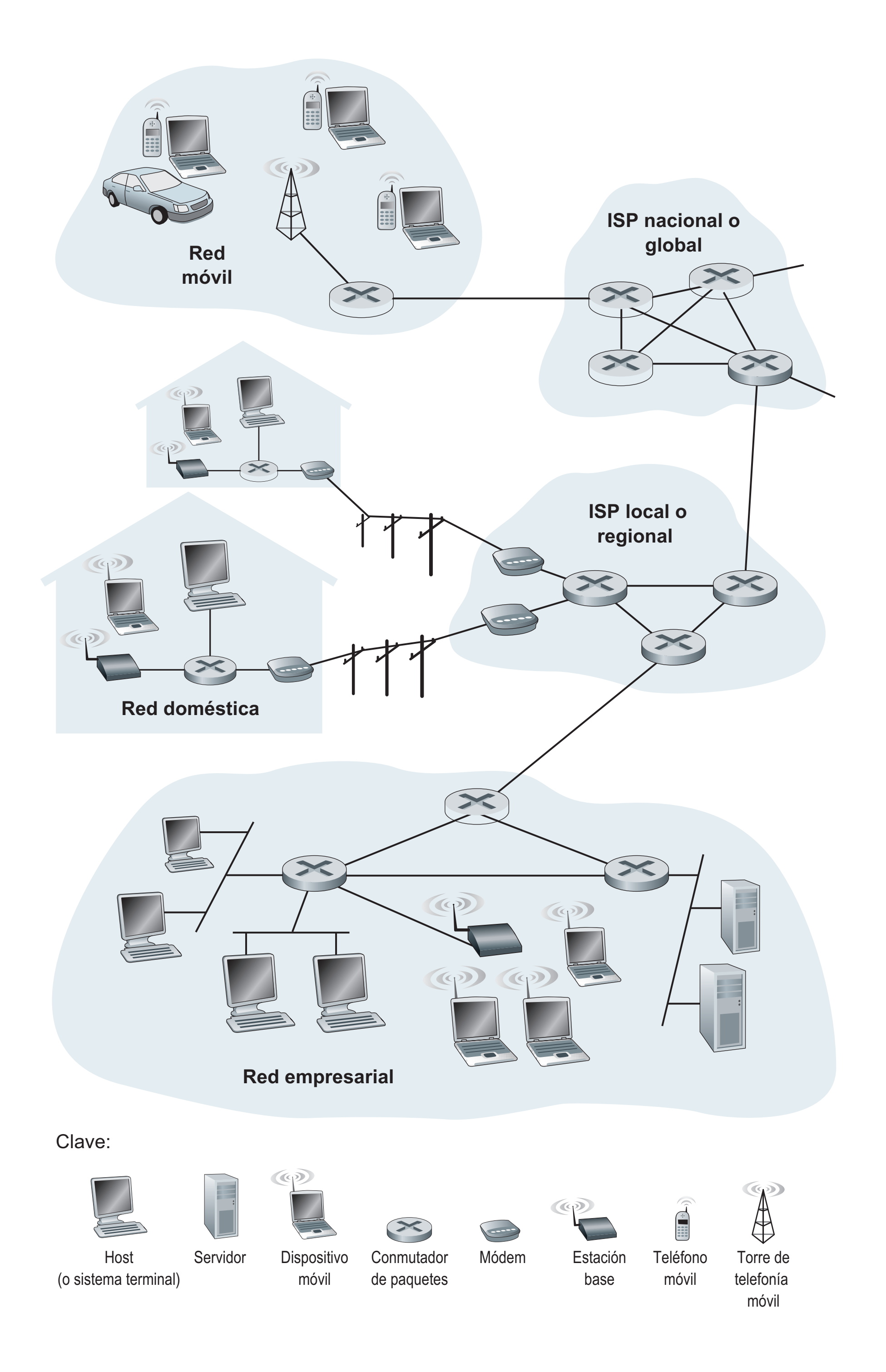


Figura 1.1 • Algunos de los componentes esenciales de Internet.

tipos de medios físicos, entre los que se incluyen el cable coaxial, el hilo de cobre, la fibra óptica y el espectro de radio. Los distintos enlaces pueden transmitir los datos a distintas velocidades y la **velocidad de transmisión** de un enlace se mide en bits/segundo. Cuando un sistema terminal tiene que enviar datos a otro sistema terminal, el emisor segmenta los datos y añade bits de cabecera a cada segmento. Los paquetes de información resultantes, conocidos como **paquetes** en la jerga informática, se envían entonces a través de la red hasta el sistema terminal receptor, donde vuelven a ser ensamblados para obtener los datos originales.

Un conmutador de paquetes toma el paquete que llega de uno de sus enlaces de comunicaciones de entrada y lo reenvía a uno de sus enlaces de comunicaciones de salida. Los dispositivos de conmutación de paquetes se suministran en muchas formas y modelos, pero los dos tipos más utilizados actualmente en Internet son los **routers** y los **switches de la capa de enlace**. Ambos tipos reenvían los paquetes hacia sus destinos finales. Los switches de la capa de enlace normalmente se emplean en las redes de acceso, mientras que los routers suelen utilizarse en el núcleo de la red. La secuencia de enlaces de comunicaciones y conmutadores de paquetes que atraviesa un paquete desde el sistema terminal emisor hasta el sistema terminal receptor se conoce como **ruta** a través de la red. Es difícil estimar la cantidad exacta de tráfico que se transporta a través de Internet [Odylsko 2003]. Pri-Metrica [PriMetrica 2009] estima que, en 2008, los proveedores de Internet emplearon 10 terabits por segundo de capacidad internacional y que dicha capacidad se duplica aproximadamente cada dos años.

Las redes de conmutación de paquetes (que transportan paquetes) son similares en muchos aspectos a las redes de transporte formadas por autopistas, carreteras e intersecciones (que transportan vehículos). Por ejemplo, imagine que una fábrica necesita trasladar un enorme cargamento a un cierto almacén de destino que se encuentra a miles de kilómetros. En la fábrica, el cargamento se reparte y se carga en una flota de camiones. Cada camión hace el viaje hasta el almacén de destino de forma independiente a través de la red de autopistas, carreteras e intersecciones. En el almacén de destino, la carga de cada camión se descarga y se agrupa con el resto del cargamento a medida que va llegando. Luego, en cierto sentido, los paquetes son como los camiones, los enlaces de comunicaciones como las autopistas y carreteras, los dispositivos de conmutación de paquetes como las intersecciones y los sistemas terminales son como los edificios (la fábrica y el almacén). Al igual que un camión sigue una ruta a través de la red de transporte por carretera, un paquete sigue una ruta a través de una red de computadoras.

Los sistemas terminales acceden a Internet a través de los **ISP (*Internet Service Provider*, Proveedor de servicios de Internet)**, incluyendo los ISP residenciales como son las compañías telefónicas o de cable locales; los ISP corporativos; los ISP universitarios y los ISP que proporcionan acceso inalámbrico (WiFi) en aeropuertos, hoteles, cafés y otros lugares públicos. Cada ISP es en sí mismo una red de conmutadores de paquetes y enlaces de comunicaciones. Los ISP proporcionan una amplia variedad de tipos de acceso a red a los sistemas terminales, entre los que se incluyen el acceso a través de módem de acceso telefónico a 56 kbps, el acceso de banda ancha residencial, mediante módem por cable o DSL, el acceso LAN (*Local Area Network*, Red de área local) de alta velocidad y el acceso inalámbrico. Los ISP también proporcionan acceso a Internet a los proveedores de contenido, conectando sitios web directamente a Internet. Internet es todo lo que conecta a los sistemas terminales entre sí, por lo que los ISP que proporcionan el acceso a los sistemas terminales también tienen que estar interconectados entre ellos. Estos ISP de nivel inferior se interco-

nectan a través de los ISP de nivel superior nacionales e internacionales, como AT&T y Sprint. Un ISP de nivel superior consiste en routers de alta velocidad interconectados a través de enlaces de fibra óptica de alta velocidad. La red de cada ISP, sea de nivel inferior o superior, se administra de forma independiente, ejecuta el protocolo IP (véase más adelante) y se ajusta a determinados convenios de denominación y de asignación de direcciones. En la Sección 1.3 examinaremos más detalladamente los ISP y sus interconexiones.

Los sistemas terminales, los conmutadores de paquetes y otros dispositivos de Internet ejecutan **protocolos** que controlan el envío y la recepción de la información dentro de Internet. El protocolo **TCP (*Transmission Control Protocol*, Protocolo de control de transmisión)** y el protocolo **IP (*Internet Protocol*, Protocolo de Internet)** son dos de los protocolos más importantes de Internet. El protocolo IP especifica el formato de los paquetes que se envían y reciben entre los routers y los sistemas terminales. Los principales protocolos de Internet se conocen colectivamente como protocolos **TCP/IP**. En este capítulo de introducción comenzaremos a estudiar los protocolos, pero esto sólo es el principio, ya que gran parte del libro se dedica a los protocolos empleados por las redes de computadoras.

Debido a la importancia de los protocolos en Internet, es importante que todo el mundo esté de acuerdo en qué hacen todos y cada uno de ellos, siendo aquí donde entran en juego los estándares. Los **estándares de Internet** son desarrollados por el IETF (*Internet Engineering Task Force*) [IETF 2009]. Los documentos asociados a estos estándares IETF se conocen como documentos **RFC (*Request For Comments*, Solicitud de comentarios)**. Los RFC nacieron como solicitudes de comentarios de carácter general (de ahí su nombre) para solucionar los problemas de diseño de la red y de los protocolos a los que se enfrentó el precursor de Internet. El contenido de estos documentos suele ser bastante técnico y detallado. Definen protocolos tales como TCP, IP, HTTP (para la Web) y SMTP (para el correo electrónico). Actualmente, existen más de 5.000 documentos RFC. Existen también otros organismos dedicados a especificar estándares para componentes de red, más específicamente para los enlaces de red. El comité de estándares IEEE 802 LAN/MAN [IEEE 802 2009], por ejemplo, especifica los estándares para redes Ethernet y WiFi.

1.1.2 Descripción de los servicios

Hasta el momento hemos identificado muchos de los componentes que forman Internet, pero también podemos describir Internet desde un punto de vista completamente diferente, en concreto como *una infraestructura que proporciona servicios a las aplicaciones.* Entre estas aplicaciones se incluyen el correo electrónico, la navegación web, la mensajería instantánea, Voz sobre IP (VoIP), la radio por Internet, los flujos de vídeo, los juegos distribuidos, la compartición de archivos en redes entre iguales o entre pares (P2P, *Peer-to-peer*), la televisión a través de Internet, las sesiones remotas y otras muchas. Se dice que las aplicaciones son **aplicaciones distribuidas**, porque implican a varios sistemas terminales que intercambian datos entre sí. Es importante saber que las aplicaciones de Internet se ejecutan en los sistemas terminales, no en los conmutadores de paquetes disponibles en el núcleo de la red. Aunque los dispositivos de conmutación de paquetes facilitan el intercambio de datos entre sistemas terminales, no se preocupan de la aplicación que esté actuando como origen o destino de los datos.

Vamos a ahondar un poco más en lo que queremos decir con una infraestructura que proporciona servicios a las aplicaciones. Para ello, supongamos que tenemos una excitante

nueva idea para una aplicación distribuida de Internet, que puede beneficiar enormemente a la humanidad o que simplemente puede hacernos ricos y famosos. ¿Cómo podríamos transformar esa idea en una aplicación real de Internet? Puesto que las aplicaciones se ejecutan en los sistemas terminales, tendremos que escribir programas software que se ejecuten en dichos sistemas. Por ejemplo, podríamos escribir programas en Java, C o Python. Ahora bien, dado que estamos desarrollando una aplicación Internet distribuida, los programas que se ejecuten en los distintos sistemas terminales tendrán que enviarse datos entre sí. Y aquí es cuando llegamos al meollo de la cuestión, a la que nos lleva a la forma alternativa de describir Internet como una plataforma para aplicaciones. ¿Cómo una aplicación que se ejecuta en un sistema terminal instruye a Internet para entregar datos a otro programa que se ejecuta en otro sistema terminal?

Los sistemas terminales conectados a Internet proporcionan una **API (*Application Programming Interface*, Interfaz de programación de aplicaciones)**, que especifica cómo un programa de software que se ejecuta en un sistema terminal pide a la infraestructura de Internet que suministre datos a un programa de software de destino específico que se ejecuta en otro sistema terminal. La API de Internet consta de un conjunto de reglas que el programa que transmite los datos debe cumplir para que Internet pueda entregar esos datos al programa de destino. En el Capítulo 2 se aborda en detalle la API de Internet. Por el momento, veamos una sencilla analogía, una que emplearemos con frecuencia a lo largo de este libro. Supongamos que Alicia desea enviar una carta a Benito utilizando el servicio postal. Por supuesto, Alicia no puede escribir la carta (los datos) y lanzar la carta por la ventana. En lugar de ello, será necesario que Alicia introduzca la carta en un sobre, escriba el nombre completo de Benito, su dirección y código postal en el sobre, lo cierre y pegue un sello en la esquina superior derecha del sobre. Por último, tendrá que introducir el sobre en un buzón del servicio postal. Por tanto, el servicio postal de correos tiene su propia “API de servicio postal”, es decir, su propio conjunto de reglas, que Alicia debe seguir para que el servicio de correos entregue su carta a Benito. De forma similar, Internet tiene una API que el programa que envía los datos debe seguir para que Internet entregue los datos al software que los recibirá.

Por supuesto, el servicio de correos proporciona más de un servicio a sus clientes, como correo urgente, acuse de recibo, correo ordinario y otros muchos. Del mismo modo, Internet proporciona múltiples servicios a sus aplicaciones. Cuando desarrolle una aplicación de Internet, también tendrá que seleccionar uno de los servicios de Internet para su aplicación. En el Capítulo 2 describiremos los servicios de Internet.

Esta segunda descripción de Internet, una infraestructura que permite proporcionar servicios a aplicaciones distribuidas, es muy importante. Cada vez más, las necesidades de las nuevas aplicaciones están dirigiendo los avances de los componentes esenciales de Internet. Por tanto, es importante tener presente que Internet es una infraestructura en la que se están inventando e implementando constantemente nuevas aplicaciones.

Aquí sólo hemos dado dos descripciones de Internet; una en términos de sus componentes esenciales y otra como infraestructura que permite proporcionar servicios a aplicaciones distribuidas. Pero es posible que todavía no tenga claro qué es Internet. ¿Qué es la conmutación de paquetes, TCP/IP y una API? ¿Qué son los routers? ¿Qué tipos de enlaces de comunicaciones existen en Internet? ¿Qué es una aplicación distribuida? ¿Cómo puede una tostadora o un sensor de temperatura conectarse a Internet? Si se siente un poco abrumado por todas estas preguntas, no se preocupe, el propósito de este libro es presentarle tanto los componentes hardware como software de Internet, así como los principios que regulan cómo y por qué funciona. En las siguientes secciones y capítulos explicaremos todos estos términos y daremos respuesta a estas cuestiones.

1.1.3 ¿Qué es un protocolo?

Ahora que ya hemos visto por encima para qué sirve Internet, vamos a ocuparnos de otro término importante en el mundo de las redes de computadoras: *protocolo*. ¿Qué es un protocolo? ¿Qué hace un protocolo*?*

Analogía humana

Probablemente, sea más sencillo comprender el concepto de protocolo de red considerando en primer lugar algunas analogías humanas, ya que las personas llevamos a cabo protocolos casi constantemente. Piense en lo que hace cuando necesita preguntar a alguien qué hora es. En la Figura 1.2 se muestra cómo se lleva a cabo un intercambio de este tipo. El protocolo entre personas (o las buenas maneras, al menos) dicta que para iniciar un proceso de comunicación con alguien lo primero es saludar (el primer "Hola" mostrado en la Figura 1.2). La respuesta típica a este saludo será también "Hola". Implícitamente, el saludo de respuesta se toma como una indicación de que se puede continuar con el proceso de comunicación y preguntar la hora. Una respuesta diferente al "hola" inicial (como por ejemplo, ¡No me moleste" o "No hablo su idioma", o cualquier otra respuesta impublicable no debemos escribir) indicará una indisposición o incapacidad para comunicarse. En este caso, el protocolo de las relaciones entre personas establece que no debe preguntarse la hora. En ocasiones, no se obtiene ninguna respuesta, en cuyo caso habrá que renunciar a preguntar a esa persona la hora que es. Tenga en cuenta que, en el protocolo entre personas, *existen mensajes específicos que*

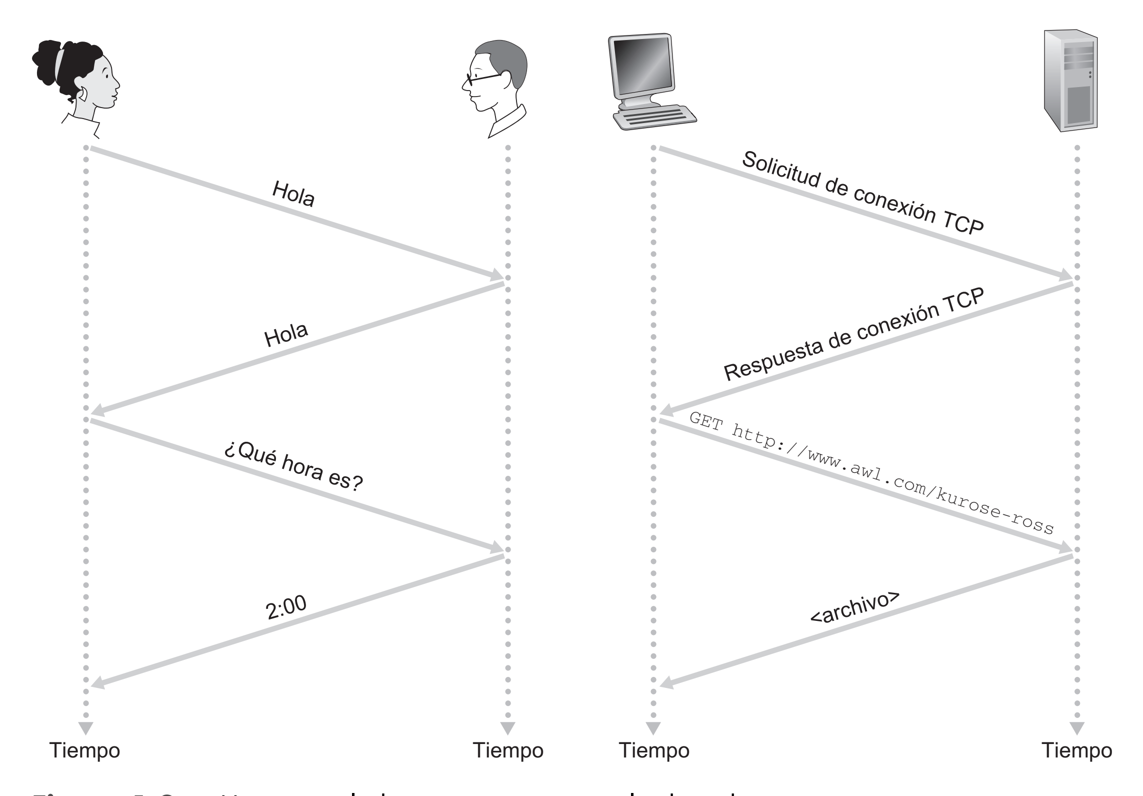


Figura 1.2 • Un protocolo humano y un protocolo de red.

enviamos y acciones específicas que tomamos como respuesta a los mensajes de contestación recibidos o a otros sucesos (como por ejemplo no recibir una respuesta en un periodo de tiempo determinado). Evidentemente, los mensajes transmitidos y recibidos y las acciones tomadas al enviar o recibir estos mensajes u otros sucesos desempeñan un papel principal en el protocolo humano. Si las personas adoptan protocolos diferentes (por ejemplo, si una persona guarda las formas pero la otra no lo hace, o si uno comprende el concepto de tiempo y el otro no), los protocolos no interoperarán y la comunicación no podrá tener lugar. Esta misma idea también es aplicable a las redes: es necesario que las entidades (dos o más) que deseen comunicarse ejecuten el mismo protocolo para poder llevar a cabo la tarea.

Consideremos ahora una segunda analogía humana. Suponga que está asistiendo a una clase (por ejemplo, sobre redes). El profesor está hablando acerca de los protocolos y usted no le comprende. El profesor detiene su explicación y dice: “¿Alguna pregunta?” (un mensaje dirigido a todos los estudiantes, que no están dormidos, y que todos ellos reciben). Usted levanta la mano (transmitiendo un mensaje implícito al profesor). El profesor le dirige una sonrisa y le dice “¿Si . . .?” (mensaje que le anima a plantear su pregunta, ya que los profesores *adoran* que les planteen cuestiones) y, a continuación, usted hace la pregunta (es decir, transmite su mensaje al profesor). El profesor escucha la pregunta (recibe su mensaje) y le responde (le transmite una respuesta). De nuevo, vemos que la transmisión y la recepción de mensajes y el conjunto de acciones convencionales tomadas cuando se envían y reciben estos mensajes, constituyen el núcleo de este protocolo de pregunta-respuesta.

Protocolos de red

Un protocolo de red es similar a un protocolo humano, excepto en que las entidades que intercambian mensajes y llevan a cabo las acciones son los componentes hardware o software de cierto dispositivo (por ejemplo, una computadora, una PDA, un teléfono móvil, un router u otro dispositivo de red). Cualquier actividad de Internet que implique dos o más entidades remotas que se comunican está gobernada por un protocolo. Por ejemplo, los protocolos implementados por hardware en las tarjetas de interfaz de red de dos computadoras conectadas físicamente controlan el flujo de bits a través del “cable” conectado entre las dos tarjetas de interfaz de red; los protocolos de control de congestión de los sistemas terminales controlan la velocidad a la que se transmiten los paquetes entre el emisor y el receptor; los protocolos de los routers determinan la ruta que seguirá un paquete desde el origen al destino. Los protocolos se ejecutan por todas partes en Internet y, en consecuencia, gran parte de este libro está dedicada a los protocolos de redes de computadoras.

Basándonos en un protocolo de red con el que probablemente estará familiarizado, vamos a ver lo que ocurre cuando se hace una solicitud a un servidor web, es decir, cuando usted escribe el URL de una página web en un navegador. Este escenario se ilustra en la mitad derecha de la Figura 1.2. En primer lugar, su computadora enviará un mensaje de solicitud de conexión al servidor web y esperará una respuesta. El servidor web recibirá su mensaje de solicitud y le devolverá un mensaje de respuesta de conexión. Sabiendo ahora que es posible solicitar el documento web, su computadora envía el nombre de la página web que desea extraer del servidor web mediante un mensaje GET. Por último, el servidor web envía la página web (archivo) a su computadora.

Basándonos en los ejemplos anteriores de protocolos humanos y de red, el intercambio de mensajes y las acciones tomadas cuando se envían y reciben estos mensajes constituyen los elementos claves para la definición de un protocolo:

Un **protocolo** *define el formato y el orden de los mensajes intercambiados entre dos o más entidades que se comunican, así como las acciones tomadas en la transmisión y/o la recepción de un mensaje u otro suceso.*

Internet, y las redes de computadoras en general, hacen un uso extensivo de los protocolos. Los distintos protocolos se utilizan para llevar a cabo las distintas tareas de comunicación. Como podrá leer a lo largo del libro, verá que algunos protocolos son simples y directos, mientras que otros son complejos e intelectualmente profundos. Dominar el campo de las redes de computadoras es equivalente a entender el qué, el por qué y el cómo de los protocolos de red.

1.2 La frontera de la red

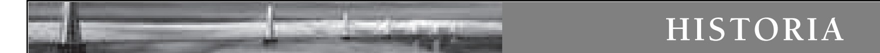
En la sección anterior hemos presentado una introducción de carácter general sobre Internet y los protocolos de red. Ahora vamos a profundizar un poco más en los componentes de una red de computadoras (y de Internet, en concreto). Comenzaremos la sección en la frontera de una red y nos fijaremos en los componentes con los que estamos más familiarizados, es decir, las computadoras, las PDA, los teléfonos móviles y otros dispositivos que utilizamos a diario. En la siguiente sección nos desplazaremos desde la frontera de la red hasta el núcleo de la misma y examinaremos los procesos de conmutación y enrutamiento que tienen lugar en las redes.

Recuerde de la sección anterior que en la jerga de las redes informáticas, las computadoras y el resto de los dispositivos conectados a Internet a menudo se designan como sistemas terminales, porque se sitúan en la frontera de Internet, como se muestra en la Figura 1.3. Entre los sistemas terminales de Internet se incluyen las computadoras de escritorio (por ejemplo, PC de escritorio, computadoras Mac y equipos Linux), servidores (por ejemplo, servidores web y de correo electrónico) y equipos móviles (por ejemplo, computadoras portátiles, dispositivos PDA y teléfonos con conexiones a Internet inalámbricas). Además, una cantidad creciente de dispositivos alternativos están actualmente conectándose a Internet como sistemas terminales (véase el recuadro de la página siguiente).

Los sistemas terminales también se conocen como *hosts,* ya que albergan (es decir, ejecutan) programas de aplicación tales como navegadores web, servidores web, programas de lectura de mensajes de correo electrónico o servidores de correo electrónico. A lo largo de este libro utilizaremos indistintamente los términos host y sistema terminal; es decir, *host* = *sistema terminal*. En ocasiones, los hosts se clasifican en dos categorías: **clientes** y **servidores**. En general, podríamos decir que los clientes suelen ser las computadoras de escritorio y portátiles, las PDA, etc., mientras que los servidores suelen ser equipos más potentes que almacenan y distribuyen páginas web, flujos de vídeo, correo electrónico, etc.

1.2.1 Programas cliente y servidor

En el contexto del software de red, existe otra definición para los términos cliente y servidor, definición a la que haremos referencia a lo largo del libro. Un **programa cliente** es un programa que se ejecuta en un sistema terminal que solicita y recibe un servicio de un **programa servidor** que se ejecuta en otro sistema terminal. La Web, el correo electrónico, la



HISTORIA

UNA ASOMBROSA COLECCIÓN DE SISTEMAS TERMINALES DE INTERNET

No hace demasiado tiempo, los sistemas terminales conectados a Internet eran fundamentalmente computadoras tradicionales como los equipos de escritorio y los potentes servidores. Desde finales de la década de 1990 y hasta el momento actual, un amplio rango de interesantes dispositivos cada vez más diversos están conectándose a Internet. Todos estos dispositivos comparten la característica común de necesitar enviar y recibir datos digitales hacia y desde otros dispositivos. Dada la omnipresencia de Internet, sus protocolos bien definidos (estandarizados) y la disponibilidad de productos hardware preparados para Internet, lo lógico es utilizar la tecnología de Internet para conectar estos dispositivos entre sí.

Algunos de estos dispositivos parecen haber sido creados exclusivamente para el entretenimiento. Un marco de fotografías IP de escritorio [Ceiva 2009] descarga fotografías digitales de un servidor remoto y las muestra en un dispositivo que parece un marco para fotografías tradicional; una tostadora Internet descarga información metereológica de un servidor y graba una imagen de la previsión del día (por ejemplo, nubes y claros) en su tostada matutina [BBC 2001]. Otros dispositivos proporcionan información útil; por ejemplo, las cámaras web muestran el estado del tráfico y las condiciones meteorológicas o vigilan un lugar de interés, los electrodomésticos conectados a Internet, entre los que se incluyen lavadoras, frigoríficos y hornos, incorporan interfaces de tipo navegador web que permiten su monitorización y control remotos. Los telefónos móviles IP con capacidades GPS (como el nuevo iPhone de Apple) ponen al alcance de la mano la navegación por la Web, el uso del correo electrónico y de servicios dependientes de la ubicación. Una nueva clase de sistemas de sensores de red promete revolucionar la forma en que observaremos e interactuaremos con nuestro entorno. Los sensores en red integrados en nuestro entorno físico permiten la vigilancia de edificios, puentes, de la actividad sísmica, de hábitats de la fauna y la flora, de estuarios y de las capas inferiores de la atmósfera [CENS 2009, CASA 2009]. Los dispositivos biomédicos pueden estar integrados y conectados en red, dando lugar a numerosos problemas de seguridad e intimidad [Halperin 2008]. Un transpondedor RFID (identificación por radiofrecuencia) o un pequeño sensor integrado en cualquier objeto puede hacer que la información acerca del objeto esté disponible en Internet, lo que nos permitirá disfrutar de una "Internet de objetos" [ITU 2005].

transferencia de archivos, las sesiones remotas, los grupos de noticias y muchas otras aplicaciones populares adoptan el modelo cliente-servidor. Puesto que un programa cliente normalmente se ejecuta en una computadora y el programa servidor en otra, las aplicaciones Internet cliente-servidor son, por definición, **aplicaciones distribuidas**. El programa cliente y el programa servidor interactúan enviándose entre sí mensajes a través de Internet. En este nivel de abstracción, los routers, los enlaces y los restantes componentes de Internet sirven de forma colectiva como una caja negra que transfiere mensajes entre los componentes distribuidos entre los que se establece la comunicación de una aplicación de Internet. Este nivel de abstracción se ilustra en la Figura 1.3.

No todas las aplicaciones de Internet actuales están constituidas por programas cliente puros que interactúan con programas servidor puros. Cada vez más aplicaciones son aplica-

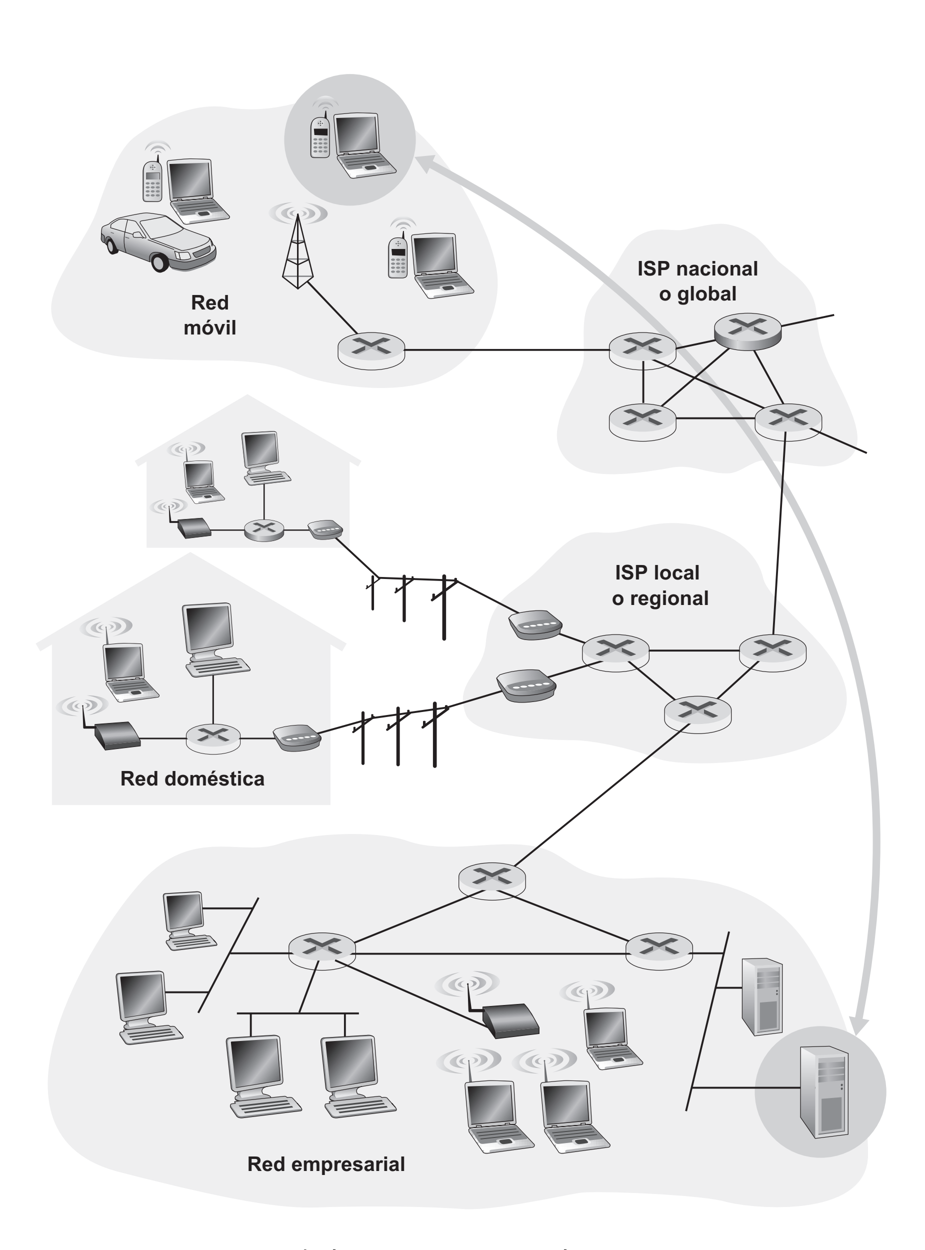


Figura 1.3 • Interacción de los sistemas terminales.

ciones entre iguales o entre pares (P2P, *Peer-to-Peer*), en las que los sistemas terminales interactúan y ejecutan programas que realizan tanto funciones de cliente como de servidor. Por ejemplo, en las aplicaciones de compartición de archivos P2P (como BitTorrent y eMule), el programa disponible en el sistema terminal del usuario actúa como cliente cuando solicita un archivo a un par y como servidor cuando envía un archivo a otro par. En la telefonía por Internet, las dos partes que intervienen en la comunicación interactúan como iguales (la sesión es simétrica, enviando y recibiendo ambas partes datos). En el Capítulo 2, compararemos y contrastaremos en detalle las arquitecturas cliente-servidor y P2P.

1.2.2 Redes de acceso

Una vez vistas las aplicaciones y los sistemas terminales existentes en la "frontera de la red", podemos pasar a ver las redes de acceso, los enlaces físicos que conectan un sistema terminal con el primer router (conocido también como "router de frontera") de una ruta entre el sistema terminal y cualquier otro sistema terminal distante. La Figura 1.4 muestra varios tipos de enlaces de acceso entre un sistema terminal y el router de frontera (los enlaces de acceso están resaltados mediante líneas más gruesas). En esta sección se repasan muchas de

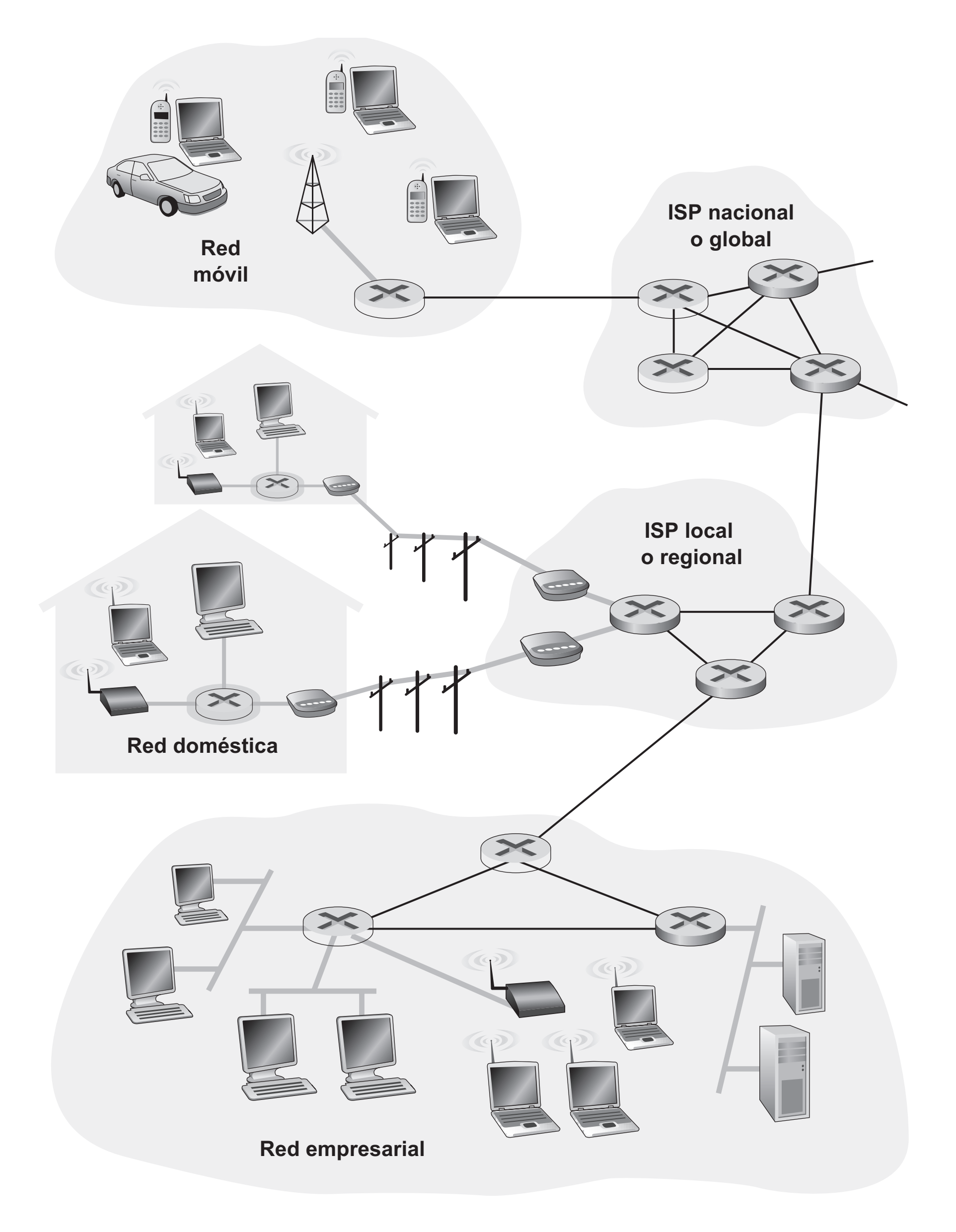


Figura 1.4 • Redes de acceso.

las tecnologías más comunes de las redes de acceso, desde las de baja velocidad hasta las de alta velocidad.

Enseguida veremos que muchas de estas tecnologías de acceso emplean, en distintos grados, partes de la infraestructura de la telefonía cableada tradicional local, la cual es proporcionada por la compañía de telefonía local, a la que haremos referencia simplemente como **telco** local. Algunos ejemplos de estas compañías serían Verizon en Estados Unidos y France Telecom en Francia. Cada residencia (chalet o piso) dispone de un enlace directo de cobre de par trenzado a un switch de la compañía telefónica, el cual se encuentra en un edificio denominado **central telefónica** en la jerga del campo de la telefonía. (Más adelante en esta sección explicaremos lo que es un cable de cobre de par trenzado.) Normalmente, una compañía telefónica local posee cientos de centrales telefónicas y enlaza a cada uno de sus clientes con la central más próxima.

Acceso telefónico

En la década de 1990, casi todos los usuarios residenciales accedían a Internet a través de las líneas telefónicas analógicas normales utilizando un módem de acceso telefónico. Actualmente, muchos usuarios de países subdesarrollados y de áreas rurales en países desarrollados (donde el acceso de banda ancha no está disponible) todavía tienen que acceder a Internet mediante una conexión de acceso telefónico. De hecho, se estima que el 10% de los usuarios residenciales de Estados Unidos utilizaban en 2008 conexiones de acceso telefónico [Pew 2008].

Se utiliza el término "acceso telefónico" (*dial-up*) porque el software del usuario realmente llama al número de teléfono de un ISP y establece una conexión telefónica tradicional con el mismo (por ejemplo, con AOL). Como se muestra en la Figura 1.5, el PC está conectado a un módem de acceso telefónico, que a su vez está conectado a la línea telefónica analógica del domicilio. Esta línea telefónica analógica está hecha de un hilo de cobre de paz trenzado y es la misma línea de teléfono que se emplea para las llamadas telefónicas ordinarias. El módem convierte la salida digital del PC en una señal analógica apropiada para ser transmitida a través de la línea telefónica analógica. En el otro extremo de la conexión, un módem del ISP convierte la señal analógica que recibe en una señal digital que será la señal de entrada para el router del ISP.

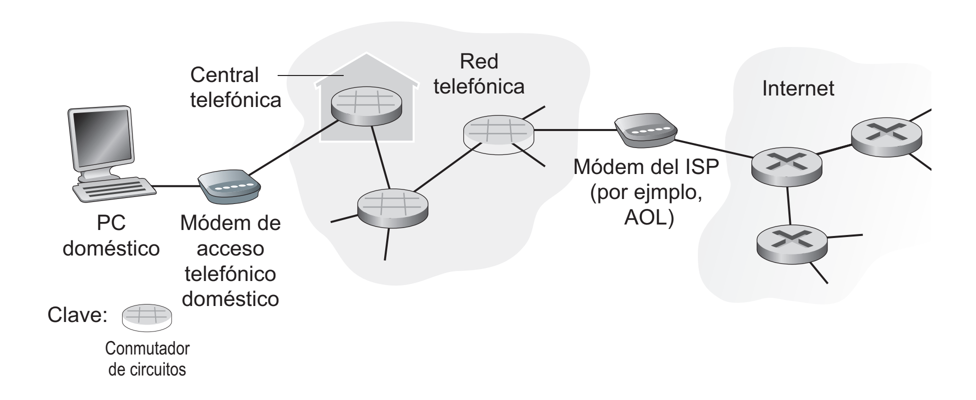


Figura 1.5 • Acceso telefónico a Internet.

El acceso telefónico a Internet presenta dos inconvenientes importantes. El primero y más destacable es que es extremadamente lento, proporcionando una velocidad máxima de 56 kbps. A esta velocidad, se tardan aproximadamente ocho minutos en descargar un archivo MP3 de una canción de tres minutos y se necesitarían varios días para descargar una película de 1 Gbyte. El segundo, un módem de acceso telefónico ocupa la línea telefónica del usuario, así que mientras que un miembro de la familia utiliza el módem de acceso telefónico para navegar por la Web, el resto de la familia no puede recibir ni hacer llamadas telefónicas normales a través de esa línea de teléfono.

DSL

Hoy en día, los dos tipos de acceso residencial de banda ancha predominantes son las líneas DSL (*Digital Subscriber Line*, Línea de abonado digital) y el cable. En la mayoría de los países desarrollados de hoy en día, más del 50% de los domicilios particulares disponen de acceso de banda ancha, con Corea del Sur, Islandia, Holanda, Dinamarca y Suiza a la cabeza con una penetración de más del 74% de los hogares en 2008 [ITIF 2008]. En Estados Unidos, las líneas DSL y cable tienen aproximadamente la misma cuota de mercado para el acceso de banda ancha [Pew 2008]. Fuera de Estados Unidos y Canadá domina la tecnología DSL, especialmente en Europa, donde en muchos países más del 90% de las conexiones de banda ancha se hacen mediante DSL.

Por regla general, los domicilios particulares contratan el servicio DSL de acceso a Internet con la misma empresa que le proporciona el acceso telefónico local (es decir, la compañía telefónica). Por tanto, cuando se utiliza el acceso mediante DSL, la compañía telefónica del cliente también actúa como ISP. Como se muestra en la Figura 1.6, cada módem DSL de un cliente utiliza la línea telefónica existente (hilo de cobre de par trenzado) para intercambiar datos con un multiplexor de acceso DSL (DSLAM), que normalmente se encuentra en la central de la compañía telefónica. La línea telefónica transporta simultáneamente los datos y las señales telefónicas, las cuales se codifican a frecuencias distintas:

- Un canal de descarga de alta velocidad opera en la banda de 50 kHz a 1 MHz.
- Un canal de carga de velocidad media opera en la banda de 4 kHz a 50 kHz.
- Un canal telefónico ordinario bidireccional opera en la banda de 0 a 4 kHz.

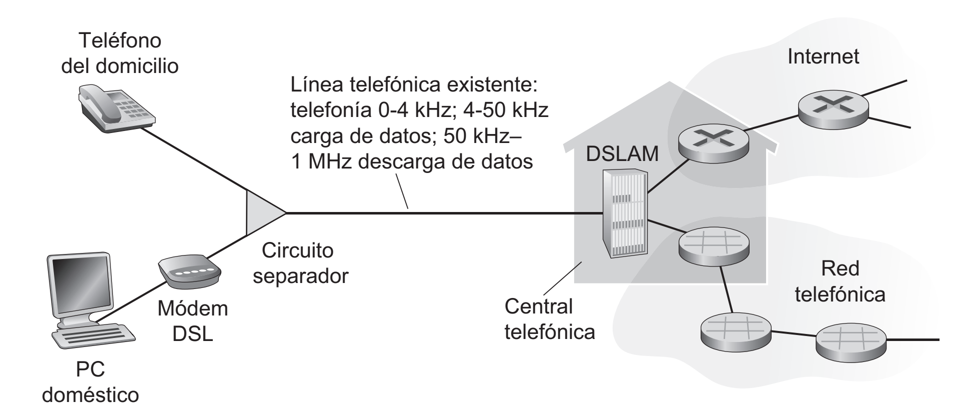


Figura 1.6 • Acceso mediante DSL a Internet.

Este método hace que un único enlace DSL se comporte como tres enlaces separados, de manera que una llamada de teléfono y una conexión a Internet pueden compartir el enlace DSL a un mismo tiempo. En la Sección 1.3.1 describiremos esta técnica de multiplexación por división en frecuencia. En el lado del cliente, las señales que llegan al domicilio son separadas como señales de datos y telefónicas mediante un circuito separador (*splitter*) que reenvía la señal de datos al módem DSL. En el lado de la compañía telefónica, en la central, el multiplexor DSLAM separa las señales de datos y de telefonía y envía los datos a Internet. Cientos o incluso miles de viviendas se conectan a un mismo DSLAM [Cha 2009, Dischinger 2007].

DSL presenta dos ventajas principales en comparación con el método de acceso telefónico a Internet. En primer lugar, puede transmitir y recibir datos a velocidades mucho más altas. Típicamente, un cliente DSL presentará una velocidad de transmisión en el rango comprendido entre 1 y 2 Mbps para las descargas (comunicaciones desde la central al domicilio del usuario) y de entre 128 kbps a 1 Mbps para las cargas (comunicaciones desde el domicilio a la central). Puesto que las velocidades de descarga y carga son diferentes, se dice que el acceso es *asimétrico*. La segunda ventaja importante es que los usuarios pueden hablar por teléfono y acceder a Internet simultáneamente. A diferencia del método de acceso telefónico, el usuario no tiene que llamar al número de teléfono del ISP para tener acceso a Internet; en su lugar, dispone de una conexión permanente "siempre activa" con el DSLAM del ISP (y por tanto con Internet).

Las velocidades de transmisión reales de descarga y de carga disponibles en el domicilio del usuario son función de la distancia entre la casa y la central telefónica, el calibre de la línea de par trenzado y el grado de interferencia eléctrica. Los ingenieros han diseñado expresamente sistemas DSL para distancias cortas entre el domicilio y la central, lo que ha permitido conseguir velocidades de transmisión sustancialmente mayores. Para incrementar la velocidad de transmisión de los datos, el sistema DSL se basa en algoritmos avanzados de procesamiento de señales y de corrección de errores, que pueden conducir a importantes retardos de los paquetes. Sin embargo, si el domicilio no se encuentra en un radio de entre 8 y 16 kilómetros de la central, la tecnología DSL de procesamiento de las señales ya no será tan efectiva y el usuario deberá recurrir a una forma alternativa de acceso a Internet.

Actualmente, existe también una amplia variedad de tecnologías DSL de alta velocidad que gozan de aceptación en muchos países. Por ejemplo, la tecnología VDSL (*Very-high speed DSL*), con la máxima penetración hoy día en Corea del Sur y Japón, proporciona velocidades de entre 12 y 55 Mbps para las descargas y velocidades de carga comprendidas entre 1,6 y 20 Mbps [DSL 2009].

Cable

Muchos domicilios de América del Norte y de muchos otros lugares reciben cientos de canales de televisión a través de redes de cable coaxial (veremos más adelante en esta sección el cable coaxial). En un sistema de televisión por cable tradicional, el **terminal de cabecera de cable** difunde los canales de televisión a través de una red de distribución de cable coaxial y amplificadores hasta los domicilios de los usuarios.

Mientras que la DSL y el acceso telefónico emplean la infraestructura de la telefonía local existente, el acceso por cable a Internet utiliza la infraestructura de la televisión por cable existente. Las casas obtienen el acceso por cable a Internet de la misma compañía que proporciona la televisión por cable. Como se ilustra en la Figura 1.7, la fibra óptica conecta el terminal de cabecera del cable a una serie de nodos de área situados en el vecindario, a

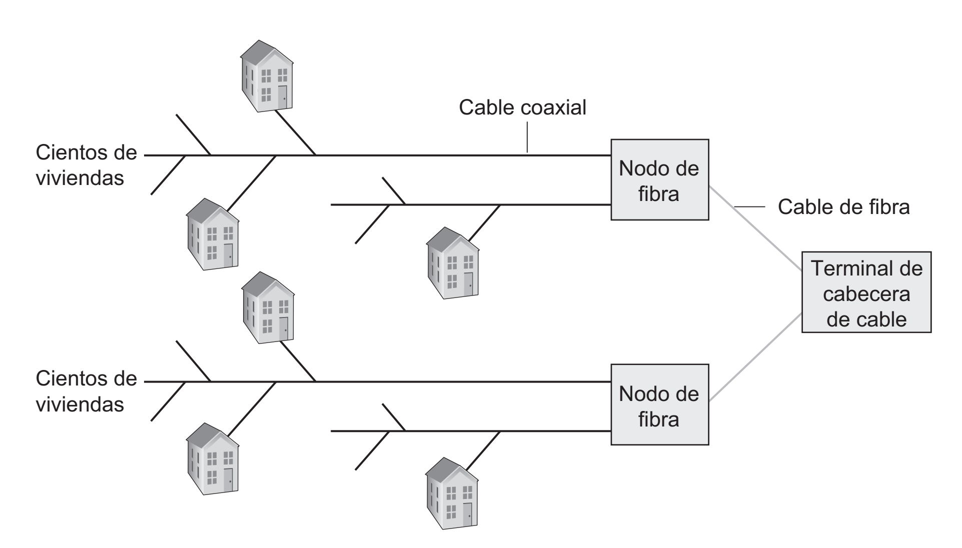


Figura 1.7 • Red de acceso híbrida de fibra óptica y cable coaxial.

partir de los cuales se utiliza el cable coaxial tradicional para llegar a todos los domicilios. Cada nodo de área suele dar soporte a entre 500 y 5.000 viviendas. Puesto que en este sistema se emplea tanto cable coaxial como fibra, a menudo se denomina sistema **HFC (*Hybrid Fiber Coax*, Híbrido de fibra y coaxial)**.

El acceso por cable a Internet requiere el uso de modems especiales, que se conocen como **modems por cable**. Al igual que un módem DSL, normalmente el módem por cable es un dispositivo externo que se conecta a un PC a través de un puerto Ethernet (en el Capítulo 5 veremos más detalles acerca de Ethernet). Los modems por cable dividen la red HFC en dos canales: un canal de descarga y un canal de carga. Al igual que en el caso de la DSL, el acceso suele ser asimétrico, teniendo normalmente el canal de descarga asignada una velocidad de transmisión mayor que el canal de carga.

Una característica importante del acceso a Internet por cable es que se trata de un medio de difusión compartido. Es decir, cada uno de los paquetes enviados por el terminal de cabecera se descargan a través de cada enlace hasta cada vivienda y los paquetes enviados desde las viviendas viajan a través del canal de carga hasta el terminal de cabecera. Así, si varios usuarios descargan simultáneamente un archivo de vídeo a través del canal de descarga, la velocidad real a la que cada usuario recibe su archivo de vídeo será significativamente menor que la velocidad acumulada de descarga por cable. Por el contrario, si sólo hay unos pocos usuarios activos que están navegando por la Web, cada uno de ellos recibirá las páginas web a la velocidad de descarga máxima del cable, ya que los usuarios rara vez solicitarán una página web al mismo tiempo. Puesto que el canal de carga también está compartido, se necesita un protocolo distribuido de acceso múltiple para coordinar las transmisiones y evitar las colisiones (veremos el problema de las colisiones en detalle en el Capítulo 5 al abordar la tecnología Ethernet).

En favor de la tecnología DSL debemos apuntar que se trata de una conexión punto a punto entre la vivienda y el ISP y que, por tanto, toda la capacidad de transmisión del enlace DSL entre el domicilio y el ISP está dedicada en lugar de ser compartida. Sin embargo, podemos decir en favor de la transmisión por cable que una red HFC correctamente dimensionada proporciona velocidades de transmisión más altas que la DSL. Existe una batalla

feroz entre las tecnologías DSL y HFC para el acceso residencial de alta velocidad, especialmente en América del Norte. En las áreas rurales, donde no está disponible ninguna de estas tecnologías, se puede utilizar un enlace vía satélite para conectar una vivienda con Internet a velocidades superiores a 1 Mbps; StarBand y HughesNet son dos proveedores de acceso vía satélite.

Tecnología FTTH (Fiber-To-The-Home, Fibra hasta el hogar)

La fibra óptica (que veremos en la Sección 1.2.3) puede ofrecer velocidades de transmisión significativamente más altas que el cable de cobre de par trenzado o el cable coaxial. En muchos países, algunas compañías telefónicas han tendido recientemente conexiones de fibra óptica desde sus centrales hasta las viviendas, proporcionando acceso a Internet de alta velocidad, así como servicios de telefonía y televisión por fibra óptica. En Estados Unidos, Verizon ha sido especialmente agresiva en el mercado de la tecnología FTTH, a través de su servicio FIOS [Verizon FIOS 2009].

Existen varias tecnologías que compiten por la distribución a través de fibra óptica desde las centrales a los hogares. La red de distribución óptica más simple se denomina **fibra directa**, en la que existe una fibra que sale de la central hasta cada domicilio. Este tipo de distribución puede proporcionar un ancho de banda grande, dado que cada cliente dispone de su propia fibra dedicada todo el camino hasta la central. Sin embargo, lo más habitual es que cada fibra saliente de la central sea compartida por muchas viviendas y ésta no se divida en fibras individuales específicas del cliente hasta llegar a un punto muy próximo a las viviendas. Hay disponibles dos arquitecturas de distribución de fibra óptica que llevan a cabo esta separación: las **redes ópticas activas (AON, *Active Optical Network*)** y las **redes ópticas pasivas (PON, *Passive Optical Network*)**. Las redes AON son fundamentalmente redes Ethernet conmutadas, las cuales abordaremos en el Capítulo 5. Aquí vamos a ver brevemente las redes ópticas pasivas, que se utilizan en el servicio FIOS de Verizon. La Figura 1.8 muestra el uso de la tecnología FTTH utilizando la arquitectura de distribución PON. Cada vivienda dispone de una terminación de red óptica (ONT, *Optical Network Terminator*), que se conecta a un distribuidor del vecindario mediante un cable de fibra óptica dedicado. El distribuidor combina una cierta cantidad de viviendas (normalmente menos de 100) en un único cable de fibra óptica compartido, que se conecta a una terminación de línea óptica (OLT,

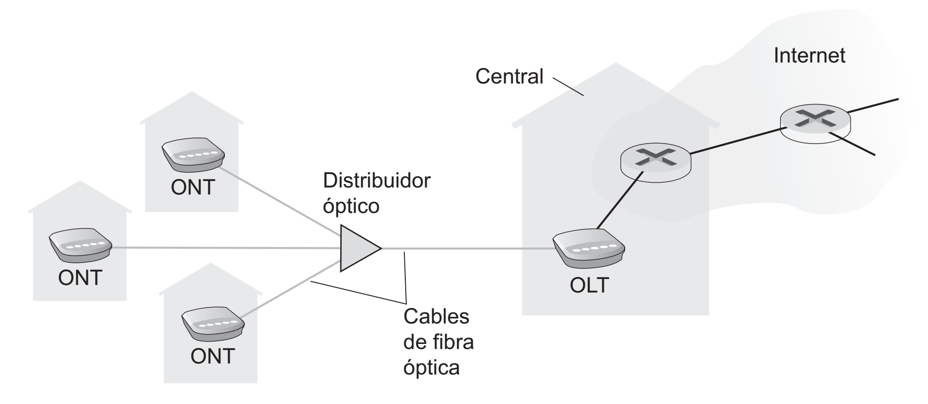


Figura 1.8 • Acceso a Internet mediante FTTH.

Optical Line Terminator) de la central de la compañía telefónica. La OLT, que realiza la conversión de señales ópticas en eléctricas, se conecta a través de Internet mediante un router de la compañía telefónica. En los domicilios, los usuarios conectan su router doméstico (normalmente un router inalámbrico) con la ONT y acceden a Internet a través de este router. En la arquitectura PON, todos los paquetes enviados desde la OLT al distribuidor se replican en este distribuidor (de forma similar a un terminal de cabecera de cable).

En teoría, la tecnología FTTH puede proporcionar velocidades de acceso a Internet del orden de los gigabits por segundo. Sin embargo, la mayoría de los ISP de FTTH ofrecen diferentes velocidades, siendo lógicamente más caras cuanto más altas son. La mayoría de los clientes actuales de la tecnología FTTH disfrutan de velocidades de descarga comprendidas entre 10 y 20 Mbps, y de velocidades de carga de entre 2 y 10 Mbps. Además del acceso a Internet, la fibra óptica permite proporcionar servicios de televisión y el servicio de telefonía tradicional.

Ethernet

En los campus universitarios y corporativos, normalmente se utiliza una red de área local (LAN, *Local Area Network*) para conectar un sistema terminal al router de frontera. Aunque existen muchos tipos de tecnologías LAN, Ethernet es con mucho la tecnología de acceso predominante en las redes corporativas y universitarias. Como se ilustra en la Figura 1.9, los usuarios de Ethernet utilizan cable de cobre de par trenzado para conectarse a un switch Ethernet (tecnología que se verá en detalle en el Capítulo 5). Con acceso Ethernet, normalmente los usuarios disponen de velocidades de acceso de 100 Mbps, y los servidores pueden alcanzar velocidades de 1 Gbps o incluso 10 Gbps.

WiFi

Cada vez es más habitual que los usuarios accedan a Internet a través de conexiones inalámbricas, bien a través de una computadora portátil o mediante un dispositivo móvil, como un iPhone, una Blackberry o un teléfono Google (véase el recuadro anterior "Una asombrosa colección de sistemas terminales de Internet"). Actualmente, existen dos tipos de acceso inalámbrico a Internet. En una **LAN inalámbrica**, los usuarios inalámbricos

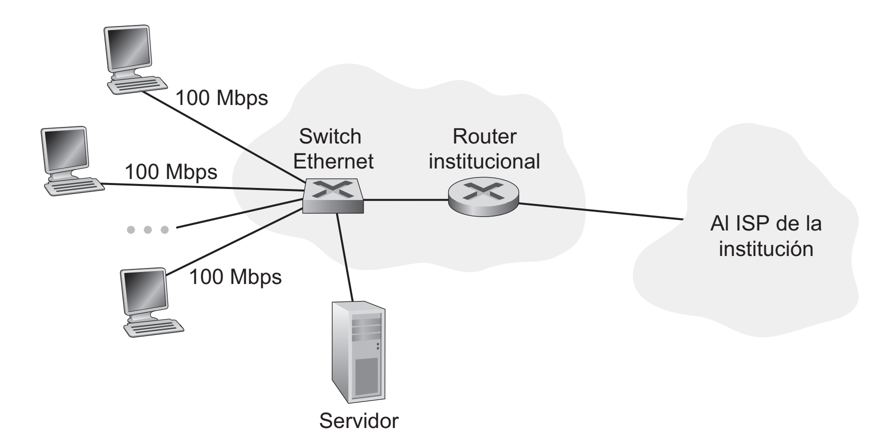


Figura 1.9 • Acceso a Internet utilizando tecnología Ethernet.

transmiten paquetes a (y reciben paquetes de) un **punto de acceso**, el cual a su vez está conectado a la red Internet cableada. Habitualmente, los usuarios de una LAN inalámbrica deben encontrarse a unas pocas decenas de metros del punto de acceso. En las **redes inalámbricas de área extensa**, los paquetes se transmiten a una **estación base** a través de la misma infraestructura inalámbrica utilizada por la telefonía móvil. En este caso, el proveedor de la red móvil gestiona la estación base y, normalmente, el usuario puede estar a unas pocas decenas de kilómetros de la estación base.

Actualmente, el acceso mediante LAN inalámbrica basada en la tecnología IEEE 802.11, es decir WiFi, podemos encontrarlo por todas partes: universidades, oficinas, cafés, aeropuertos, domicilios e incluso en los aviones. La mayor parte de las universidades han instalado estaciones base IEEE 802.11 por sus campus, lo que permite a los estudiantes enviar y recibir mensajes de correo electrónico o navegar por la Web estando en cualquier lugar del campus. En muchas ciudades, alguien puede estar parado en la esquina de una calle y encontrarse dentro del alcance de diez o veinte estaciones base (para ver un mapa global navegable de estaciones base 802.11 descubiertas y registradas en un sitio web por personas que disfrutan haciendo este tipo de cosas, consulte [wigle.net 2009]). Como se explica en el Capítulo 6, actualmente, la tecnología 802.11 proporciona una velocidad de transmisión compartida de hasta 54 Mbps.

Muchas viviendas combinan acceso residencial de banda ancha (es decir, modems por cable o DSL) con tecnología LAN inalámbrica barata para crear redes domésticas potentes. La Figura 1.10 muestra un esquema de una red doméstica típica. Esta red doméstica está formada por un portátil con función de itinerancia (*roaming*) y un PC de sobremesa; una estación base (el punto de acceso inalámbrico), que se comunica con el portátil inalámbrico; un módem por cable, que proporciona el acceso de banda ancha a Internet y un router, que interconecta la estación base y el PC de sobremesa con el módem por cable. Esta red permite a los usuarios de esta red doméstica tener acceso de banda ancha a Internet mediante un dispositivo móvil con el que se puede ir de la cocina a los dormitorios y al jardín.

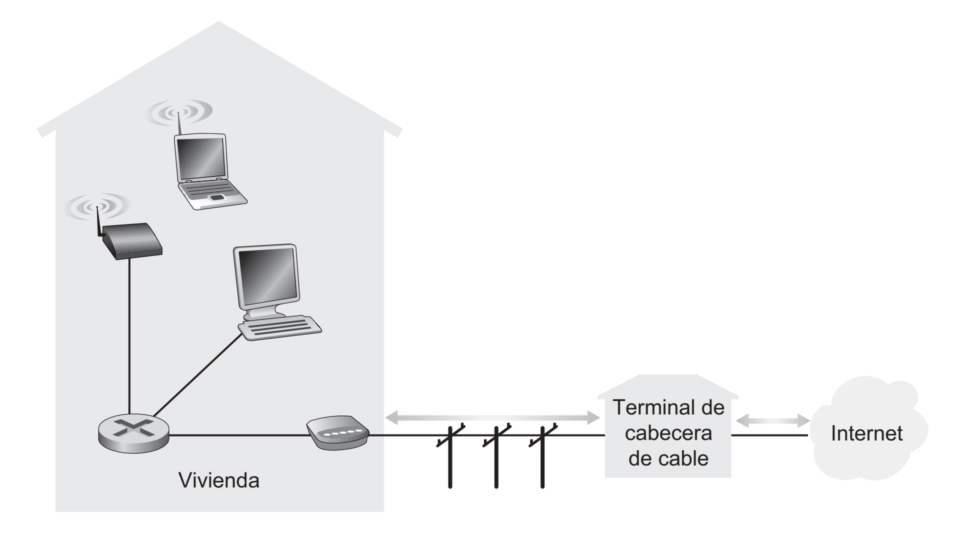


Figura 1.10 • Esquema de una red doméstica típica.

Acceso inalámbrico de área extensa

Cuando se accede a Internet a través de una red LAN inalámbrica, normalmente es necesario estar a unas pocas decenas de metros del punto de acceso. Esto es viable en viviendas, cafés y, de forma más general, en el interior y los alrededores de un edificio. Pero, ¿qué ocurre cuando se necesita tener acceso a Internet y se está en la playa, en un autobús o en el coche? Para este tipo de acceso de área extensa, los usuarios itinerantes de Internet utilizan la infraestructura de la telefonía móvil para acceder a estaciones base que están separadas entre sí unas decenas de kilómetros.

Las empresas de telecomunicaciones han hecho grandes inversiones en lo que se conoce como redes inalámbricas de tercera generación (3G), que proporcionan acceso inalámbrico a Internet mediante una red de área extensa de conmutación de paquetes a velocidades por encima de 1 Mbps. Actualmente, millones de usuarios emplean estas redes para leer y enviar mensajes de correo electrónico, navegar por la Web y descargar música mientras se desplazan de un lugar a otro.

WiMAX

Como siempre, existe una posible tecnología "definitiva" que espera destronar a estos estándares. WiMAX [Intel WiMAX 2009, WiMAX Forum 2009], también conocido como IEEE 802.16, es un primo lejano del protocolo WiFi 802.11 citado anteriormente. WiMAX opera independientemente de la red de telefonía móvil y promete velocidades comprendidas entre 5 y 10 Mbps o superiores para distancias de decenas de kilómetros. Sprint-Nextel ha invertido miles de millones de dólares en la implantación de WiMAX a partir del año 2007. En el Capítulo 6 se abordan en detalle las tecnologías WiFi, WiMAX y 3G.

1.2.3 Medios físicos

En la subsección anterior hemos proporcionado una panorámica de algunas de las tecnologías de acceso a red más importantes disponibles para Internet. Según hemos ido describiendo estas tecnologías, hemos indicado los medios físicos utilizados. Por ejemplo, hemos dicho que la tecnología HFC emplea una combinación de cable de fibra óptica y de cable coaxial. También hemos señalado que los modems de acceso telefónico a 56 kbps y las DSL utilizan cable de cobre de par trenzado. Asimismo, también hemos comentado que las redes para acceso móvil usan el espectro de radio. En esta subsección vamos a hacer una breve introducción a éstos y otros medios de transmisión que se emplean habitualmente en Internet.

Para definir lo que se entiende por medio físico, reflexionemos sobre la breve vida de un bit. Imagine un bit que viaja desde un sistema terminal atravesando una serie de enlaces y routers hasta otro sistema terminal. Este pobre bit se desplaza de un lado a otro sin descanso. En primer lugar, el sistema terminal de origen transmite el bit y poco tiempo después el primer router de la serie recibe dicho bit; el primer router transmite entonces el bit y poco después lo recibe el segundo router, y así sucesivamente. Por tanto, nuestro bit, al viajar desde el origen hasta el destino, atraviesa una serie de parejas de transmisores y receptores. En cada par transmisor-receptor, el bit se envía mediante ondas electromagnéticas o pulsos ópticos a lo largo de un **medio físico**. Este medio físico puede tener muchas formas y no tiene que ser del mismo tipo para cada par transmisor-receptor existente a lo largo de la ruta. Entre los ejemplos de medios físicos se incluyen el cable de cobre de par

trenzado, el cable coaxial, el cable de fibra óptica multimodo, el espectro de radio terrestre y el espectro de radio por satélite. Los medios físicos se pueden clasificar dentro de dos categorías: **medios guiados** y **medios no guiados**. En los medios guiados, las ondas se transportan a través de un medio sólido, como por ejemplo un cable de fibra óptica, un cable de cobre de par trenzado o un cable coaxial. En los medios no guiados, las ondas se propagan por la atmósfera y el espacio exterior, tal como ocurre en las redes LAN inalámbricas o en un canal de satélite digital.

Pero antes de abordar las características de los distintos tipos de medios, veamos algunos detalles acerca de los costes. El coste real de un enlace físico (cable de cobre, de fibra óptica, o coaxial, etc.) suele ser relativamente pequeño cuando se compara con los restantes costes de la red. En particular, el coste de mano de obra asociado con la instalación del enlace físico puede ser de varios órdenes de magnitud mayor que el coste del material. Por ello, muchos constructores instalan cables de par trenzado, de fibra óptica y coaxial en todas las habitaciones de los edificios. Incluso aunque inicialmente sólo se utilice uno de los medios, existen muchas posibilidades de que se emplee algún otro medio físico en un futuro próximo y, por tanto, se ahorre dinero al no tener que tirar cables adicionales.

Cable de cobre de par trenzado

El medio de transmisión guiado más barato y más comúnmente utilizado es el cable de cobre de par trenzado. Se ha utilizado durante un siglo en las redes telefónicas. De hecho, más del 99 por ciento de las conexiones cableadas utilizan cable de cobre de par trenzado entre el propio teléfono y el conmutador telefónico local. La mayoría de nosotros disponemos de cable de par trenzado en nuestros hogares y entornos de trabajo. Este cable consta de dos hilos de cobre aislados, de un milímetro de espesor cada uno de ellos, que siguen un patrón regular en espiral. Los hilos se trenzan para reducir las interferencias eléctricas procedentes de pares similares próximos. Normalmente, una serie de pares se meten dentro de un cable envolviendo los pares en una pantalla protectora. Un par de hilos constituyen un único enlace de comunicaciones. El **par trenzado no apantallado (UTP, *Unshielded Twisted Pair*)** se utiliza habitualmente en las redes de computadoras ubicadas dentro de un edificio, es decir, para las redes LAN. La velocidad de transmisión de datos de las LAN actuales que emplean cables de par trenzado varían entre 10 Mbps y 1 Gbps. Las velocidades de transmisión de datos que se pueden alcanzar dependen del espesor del cable y de la distancia existente entre el transmisor y el receptor.

Cuando en la década de 1980 surgió la tecnología de la fibra óptica, muchas personas despreciaron el cable de par trenzado a causa de sus relativamente bajas velocidades de transmisión. Algunos pensaron incluso que la fibra óptica desplazaría por completo al cable de par trenzado. Pero el cable de par trenzado no se daría por vencido tan fácilmente. La tecnología moderna del par trenzado, como por ejemplo los cables UTP de categoría 5, pueden alcanzar velocidades de datos de 1 Gbps para distancias de hasta 100 metros. Al final, los cables de par trenzado se han establecido como la solución dominante para las redes LAN de alta velocidad.

Como hemos mencionado anteriormente, los cables de par trenzado también suelen utilizarse para el acceso a Internet de tipo residencial. Hemos dicho que los modems de acceso telefónico permiten establecer conexiones a velocidades de hasta 56 kbps utilizando cables de par trenzado. También hemos comentado que la tecnología DSL (*Digital Subscriber Line*) ha permitido a los usuarios residenciales acceder a Internet a velocidades superiores a

6 Mbps empleando cables de par trenzado (siempre y cuando los usuarios vivan en las proximidades del módem del ISP).

Cable coaxial

Al igual que el par trenzado, el cable coaxial consta de dos conductores de cobre, pero dispuestos de forma concéntrica en lugar de en paralelo. Con esta construcción y un aislamiento y apantallamiento especiales, el cable coaxial puede proporcionar velocidades de transmisión de bit bastante altas. El cable coaxial es bastante común en los sistemas de televisión por cable. Como hemos mencionado anteriormente, recientemente los sistemas de televisión por cable han comenzado a incorporar modems por cable con el fin de proporcionar a los usuarios residenciales acceso a Internet a velocidades de 1 Mbps o superiores. En la televisión por cable y en el acceso a Internet por cable, el transmisor desplaza la señal digital a una banda de frecuencia específica y la señal analógica resultante se envía desde el transmisor a uno a o más receptores. El cable coaxial puede utilizarse como un **medio compartido** guiado; es decir, una serie de sistemas terminales pueden estar conectados directamente al cable, recibiendo todos ellos lo que envíen los otros sistemas terminales.

Fibra óptica

La fibra óptica es un medio flexible y de poco espesor que conduce pulsos de luz, representando cada pulso un bit. Un único cable de fibra óptica puede soportar velocidades de bit tremendamente altas, por encima de decenas o incluso centenas de gigabits por segundo. La fibra óptica es inmune a las interferencias electromagnéticas, presenta una atenuación de la señal muy baja hasta una distancia de 100 kilómetros y es muy difícil que alguien pueda llevar a cabo un "pinchazo" en una de estas líneas. Estas características hacen de la fibra óptica el medio de transmisión guiado a larga distancia preferido, especialmente para los enlaces transoceánicos. Muchas de las redes telefónicas para larga distancia de Estados Unidos y otros países utilizan hoy día exclusivamente fibra óptica. La fibra óptica también es el medio predominante en las redes troncales de Internet. Sin embargo, el alto coste de los dispositivos ópticos, como son los transmisores, receptores y conmutadores, están entorpeciendo su implantación para el transporte a corta distancia, como por ejemplo en el caso de una LAN o en el domicilio de una red de acceso residencial. Las velocidades del enlace estándar de portadora óptica (OC, *Optical Carrier*) están comprendidas en el rango de 51,8 Mbps a 39,8 Gbps; suele hacerse referencia a estas especificaciones como OC-*n*, donde la velocidad del enlace es igual a *n* 51,8 Mbps. Entre los estándares en uso actuales se encuentran: OC-1, OC-3, OC-12, OC-24, OC-48, OC-96, OC-192, OC-768. [IEC Optical 2009; Goralski 2001; Ramaswami 1998 y Mukherjee 1997] proporcionan información acerca de diversos aspectos de las redes ópticas.

Canales de radio terrestres

Los canales de radio transportan señales en el espectro electromagnético. Constituyen un medio atractivo porque no requieren la instalación de cables físicos, pueden atravesar las paredes, proporcionan conectividad a los usuarios móviles y, potencialmete, pueden transportar una señal a grandes distancias. Las características de un canal de radio dependen de forma significativa del entorno de propagación y de la distancia a la que la señal tenga que ser transportada. Las consideraciones ambientales determinan la pérdida del camino, la atenuación de sombra (lo que disminuye la intensidad de la señal a medida que recorre una dis-

tancia y rodea/atraviesa los objetos que obstruyen su camino), la atenuación multicamino (debida a la reflexión de la señal en los objetos que interfieren) y las interferencias (debidas a otras transmisiones y a las señales electromagnéticas).

Las canales de radio terrestre pueden clasificarse en dos amplios grupos: aquéllos que operan en las áreas locales, normalmente con un alcance de entre diez y unos cientos de metros y los que operan en un área extensa, con alcances de decenas de kilómetros. Las redes LAN inalámbricas descritas en la Sección 1.2.2 emplean canales de radio de área local y las tecnologías celulares utilizan canales de radio de área extensa. En el Capítulo 6 se estudian en detalle los canales de radio.

Canales de radio vía satélite

Las comunicaciones por satélite enlazan dos o más transmisores/receptores de microondas con base en la Tierra, que se conocen como estaciones terrestres. El satélite recibe las transmisiones en una banda de frecuencia, regenera la señal utilizando un repetidor (véase más adelante) y transmite la señal a otra frecuencia. En este tipo de comunicaciones se emplean dos tipos de satélites: los **satélites geoestacionarios** y los **satélites de la órbita baja terrestre (LEO, *Low-Earth Orbiting*)**.

Los satélites geoestacionarios están permanentemente situados en el mismo punto por encima de la Tierra. Esta presencia estacionaria se consigue poniendo el satélite en órbita a una distancia de 36.000 kilómetros por encima de la superficie de la Tierra. La distancia entre la estación terrestre y el satélite más la distancia de vuelta desde el satélite a la estación terrestre introduce un retardo de propagación de la señal de 280 milisegundos. No obstante, los enlaces vía satélite, que pueden operar a velocidades de cientos de Mbps, a menudo se emplean en áreas en las que no hay disponible acceso a Internet mediante DSL o cable.

Los satélites LEO se colocan mucho más cerca de la Tierra y no se encuentran permanentemente en la misma posición, sino que giran alrededor de la Tierra (al igual que la Luna) y pueden comunicarse entre sí, así como con las estaciones terrestres. Para poder proporcionar una cobertura continua a un área, es preciso poner en órbita muchos satélites. Actualmente se están desarrollando muchos sistemas de comunicaciones de baja altitud. La página web Lloyd's satellite constellations [Wood 2009] proporciona y recopila información acerca de los sistemas de constelaciones de satélites para comunicaciones. La tecnología de los satélites de la órbita baja terrestre (LEO) podrá utilizarse, en algún momento en el futuro, para acceder a Internet.

1.3 El núcleo de la red

Una vez que hemos examinado la frontera de Internet, vamos a adentrarnos en el núcleo de la red, la malla de conmutadores de paquetes y enlaces que interconectan los sistemas terminales de Internet. En la Figura 1.11 se ha resaltado el núcleo de la red con líneas más gruesas.

1.3.1 Conmutación de circuitos y conmutación de paquetes

Existen dos métodos fundamentales que permiten transportar los datos a través de una red de enlaces y conmutadores: la **conmutación de circuitos** y la **conmutación de paquetes**. En las redes de conmutación de circuitos, los recursos necesarios a lo largo de una ruta

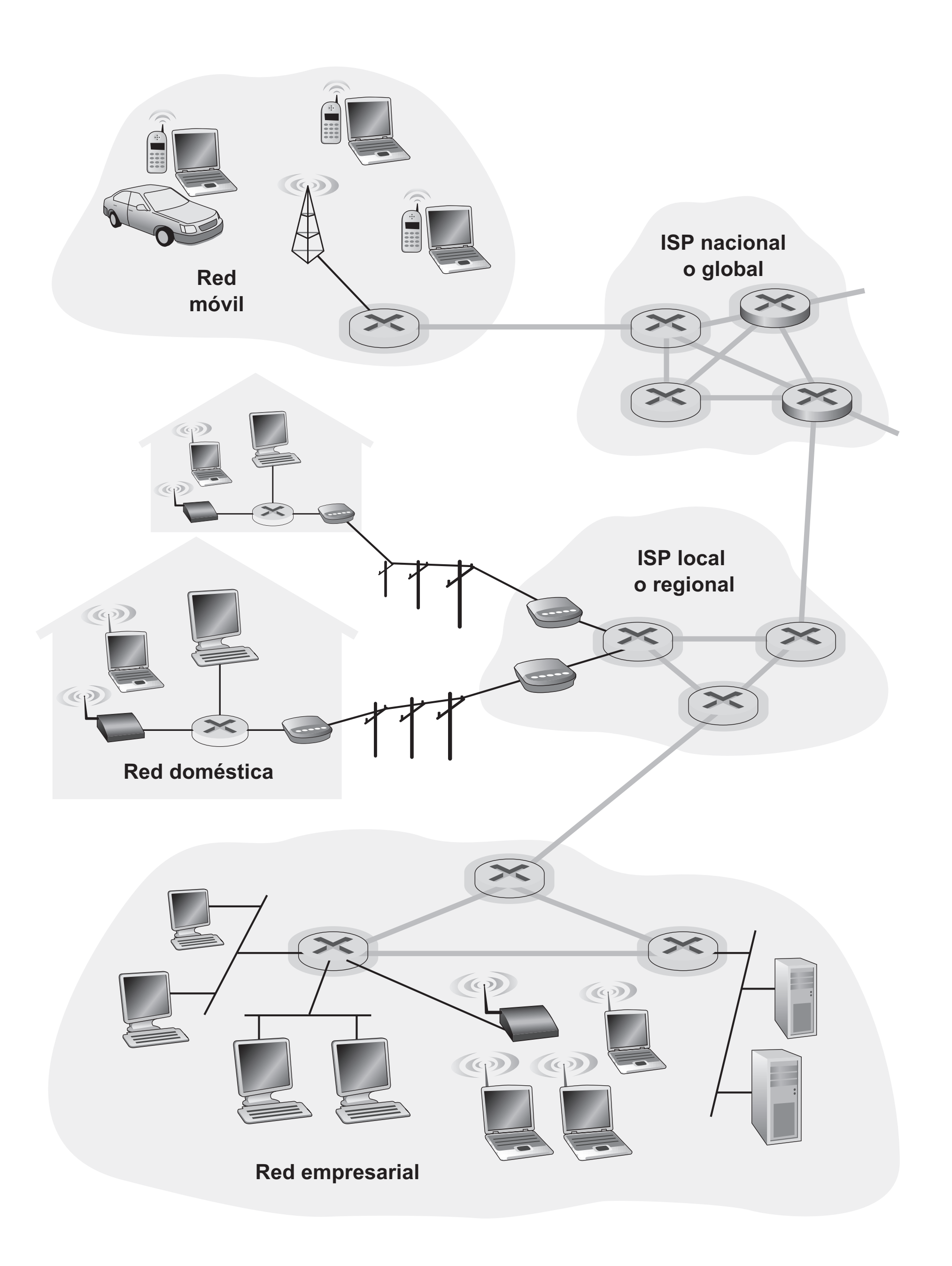


Figura 1.11 • El núcleo de la red.

(buffers, velocidad de transmisión del enlace) que permiten establecer la comunicación entre los sistemas terminales están *reservados* durante el tiempo que dura la sesión entre dichos sistemas terminales. En las redes de conmutación de paquetes, estos recursos *no* están reservados; los mensajes de una sesión utilizan los recursos bajo petición y, en con-

secuencia, pueden tener que esperar (es decir, ponerse en cola) para poder acceder a un enlace de comunicaciones. Veamos una sencilla analogía. Piense en dos restaurantes, en uno de ellos es necesario hacer reserva y en el otro no se requiere hacer reserva ni tampoco las admiten. Para comer en el restaurante que precisa reserva, tenemos que molestarnos en llamar por teléfono antes de salir de casa, pero al llegar allí, en principio, podremos sentarnos y pedir nuestro menú al camarero de manera inmediata. En el restaurante que no admite reservas, no tenemos que molestarnos en reservar mesa, pero al llegar allí, es posible que tengamos que esperar para tener una mesa antes de poder hablar con el camarero.

Las omnipresentes redes telefónicas son ejemplos de redes de conmutación de circuitos. Considere lo que ocurre cuando una persona desea enviar información (de voz o facsímil) a otra a través de una red telefónica. Antes de que el emisor pueda transmitir la información, la red debe establecer una conexión entre el emisor y el receptor. Se trata de una conexión de *buena fe* en la que los conmutadores existentes en la ruta entre el emisor y el receptor mantienen el estado de la conexión para dicha comunicación. En la jerga del campo de la telefonía, esta conexión se denomina **circuito**. Cuando la red establece el circuito, también reserva una velocidad de transmisión constante en los enlaces de la red para el tiempo que dure la conexión. Dado que el ancho de banda para esta conexión emisor-receptor ha sido reservado, el emisor puede transferir los datos al receptor a la velocidad constante *garantizada*.

La red Internet de hoy día es la quinta esencia de las redes de conmutación de paquetes. Veamos qué ocurre cuando un host desea enviar un paquete a otro host a través de Internet. Al igual que con la conmutación de circuitos, el paquete se transmite a través de una serie de enlaces de comunicaciones. Pero con la técnica de conmutación de paquetes, el paquete se envía a la red sin haber reservado ancho de banda. Si uno de los enlaces está congestionado porque otros paquetes tienen que ser transmitidos a través de él al mismo tiempo, entonces nuestro paquete tendrá que esperar en un buffer en el lado del emisor del enlace de transmisión y, por tanto, sufrirá un retardo. Internet realiza el *máximo esfuerzo* para suministrar los paquetes a tiempo, pero no existe ninguna garantía.

No todas las redes de telecomunicaciones pueden clasificarse como redes puras de conmutación de circuitos o redes puras de conmutación de paquetes. No obstante, esta clasificación es un excelente punto de partida para comprender la tecnología de las redes de telecomunicaciones.

Conmutación de circuitos

Este libro está dedicado a las redes de computadoras, Internet y la conmutación de paquetes, no a las redes telefónicas y la conmutación de circuitos. No obstante, es importante comprender por qué Internet y otras redes de computadoras utilizan la tecnología de conmutación de paquetes en lugar de la tecnología más tradicional de conmutación de circuitos de las redes telefónicas. Por esta razón, a continuación vamos a ofrecer una breve introducción a la conmutación de circuitos.

La Figura 1.12 ilustra una red de conmutación de circuitos. En esta red, los cuatro conmutadores de circuitos están interconectados mediante cuatro enlaces. Cada uno de los enlaces tiene *n* circuitos, por lo que cada enlace puede dar soporte a *n* conexiones simultáneas. Cada uno de los hosts (por ejemplo, los PC y estaciones de trabajo) está conectado directamente a uno de los conmutadores. Cuando dos hosts desean comunicarse, la red establece una **conexión terminal a terminal** dedicada entre ellos (por supuesto, las llamadas entre

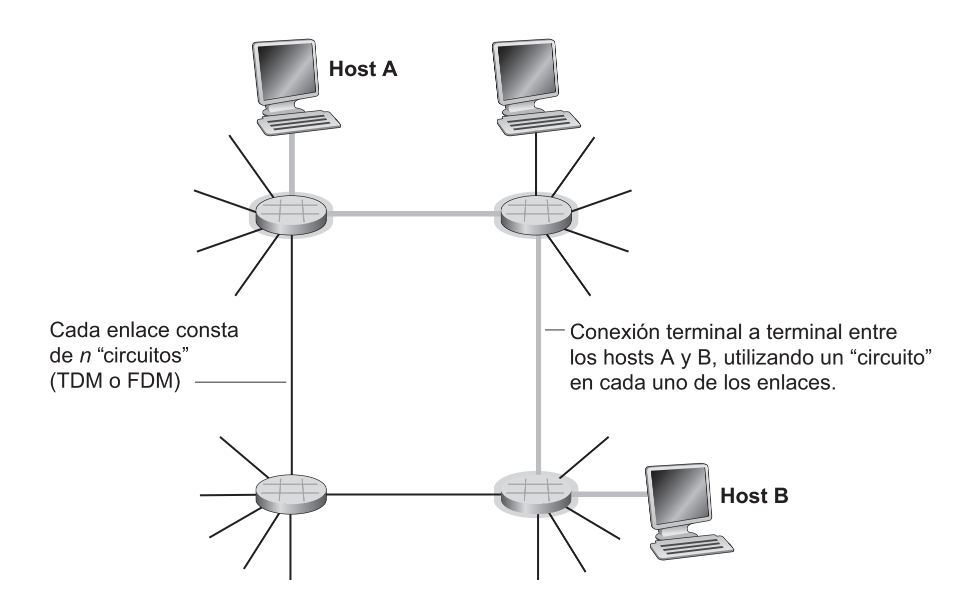


Figura 1.12 • Red de conmutación de circuitos simple formada por cuatro dispositivos de conmutación y cuatro enlaces.

más de dos dispositivos también son posibles, pero con el fin de que el lector comprenda el concepto, vamos a suponer por el momento que sólo intervienen dos hosts en cada conexión). Por tanto, para que el host A envíe mensajes al host B, la red tiene que reservar en primer lugar un circuito en cada uno de los dos enlaces. Dado que cada enlace tiene *n* circuitos, para cada enlace utilizado por la conexión terminal a terminal, la conexión obtiene una fracción 1/*n* del ancho de banda del enlace para el tiempo de duración de la conexión.

Multiplexación en redes de conmutación de circuitos

Un circuito en un enlace se implementa bien mediante **multiplexación por división de frecuencia (FDM, *Frequency-Division Multiplexing*)** o mediante **multiplexación por división en el tiempo (TDM, *Time-Division Multiplexing*)**. Con FDM, el espectro de frecuencia de un enlace se reparte entre las conexiones establecidas a lo largo del enlace. Específicamente, el enlace dedica una banda de frecuencias a cada conexión durante el tiempo que ésta dure. En las redes telefónicas, esta banda de frecuencias normalmente tiene un ancho de 4 kHz (es decir, 4.000 hercios o 4.000 ciclos por segundo). El ancho de esta banda se denomina lógicamente **ancho de banda**. Las estaciones de radio FM también emplean la multiplexación FDM para compartir el espectro de frecuencias entre 88 MHz y 108 MHz, teniendo cada estación asignada una banda de frecuencias específica.

En un enlace TDM, el tiempo se divide en marcos de duración fija y cada marco se divide en un número fijo de particiones. Cuando la red establece una conexión a través de un enlace, la red dedica una partición de cada marco a dicha conexión. Estas particiones están dedicadas para uso exclusivo de dicha conexión con una partición disponible para utilizar (en cada marco) para transmitir los datos de la conexión.

La Figura 1.13 ilustra las multiplexaciones FDM y TDM para un enlace de red específico que da soporte a cuatro circuitos. En el caso de la multiplexación por división de

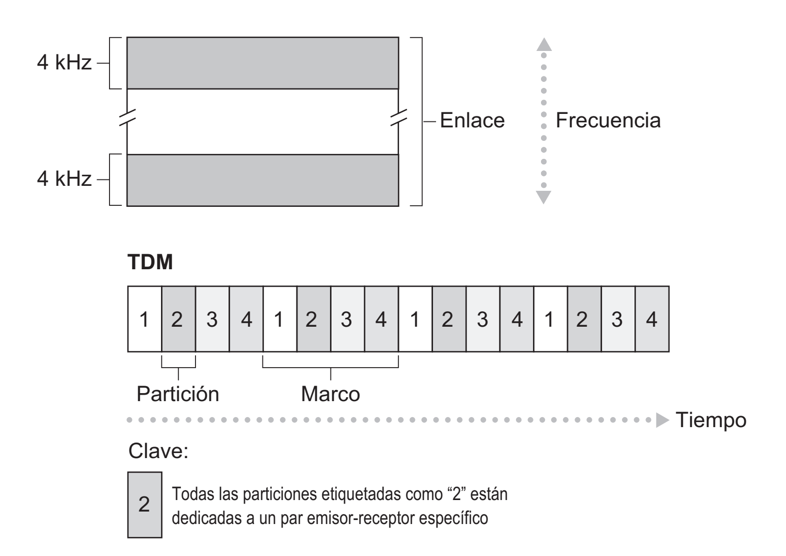


Figura 1.13 • Con FDM cada circuito obtiene de forma continua una fracción del ancho de banda. Con TDM, cada circuito dispone de todo el ancho de banda periódicamente durante breves intervalos de tiempo (es decir, durante las particiones).

frecuencia, el dominio de frecuencia se segmenta en cuatro bandas, siendo el ancho de banda de cada una de ellas de 4 kHz. En el caso de la multiplexación TDM, el dominio del tiempo se divide en cuatro marcos, conteniendo cada uno de ellos cuatro particiones. A cada circuito se le asigna la misma partición dedicada dentro de los marcos, de forma cíclica. En la multiplexación TDM, la velocidad de transmisión de un circuito es igual a la velocidad de marco multiplicada por el número de bits existentes en una partición. Por ejemplo, si el enlace transmite 8.000 marcos por segundo y cada partición consta de 8 bits, entonces la velocidad de transmisión de un circuito es igual a 64 kbps.

Los partidarios de la tecnología de conmutación de paquetes siempre han argumentado que la conmutación de circuitos es derrochadora, porque los circuitos dedicados quedan inutilizados durantes los **periodos de inactividad**. Por ejemplo, cuando una persona deja de hablar durante una llamada telefónica, los recursos de red inactivos (bandas de frecuencia o particiones temporales del enlace a lo largo de la ruta de la conexión) no pueden ser empleados por otras conexiones en curso. Otro ejemplo de cómo estos recursos pueden ser infrautilizados sería un radiólogo que empleara una red de conmutación de circuitos para acceder remotamente a una serie de radiografías de rayos X. El radiólogo establece una conexión, solicita una imagen, la contempla y luego solicita otra imagen. Los recursos de la red están asignados a la conexión pero no se utilizan (es decir, se desperdician) durante el tiempo que el radiólogo contempla las imágenes. Los partidarios de la conmutación de paquetes también disfrutan apuntando que el establecimiento de circuitos terminal a terminal y la reserva de ancho de banda terminal a terminal son procesos complicados que requieren el uso de software de señalización complejo para coordinar el funcionamiento de los switches a lo largo de la ruta terminal a terminal.

Antes de terminar con esta exposición acerca de la conmutación de circuitos, vamos a ver un ejemplo numérico que debería arrojar más luz sobre este tema. Consideremos el

tiempo que se tarda en enviar un archivo de 640.000 bits desde el host A al host B a través de una red de conmutación de circuitos. Supongamos que todos los enlaces de la red utilizan multiplexación TDM con 24 particiones y tienen una velocidad de bit de 1,536 Mbps. Supongamos también que se tardan 500 milisegundos en establecer el circuito terminal a terminal antes de que el host A pueda comenzar a transmitir el archivo. ¿Cuánto tiempo se tarda en transmitir el archivo? La velocidad de transmisión de cada circuito es (1,536 Mbps)/24 = 64 kbps, por lo que se precisan (640.000 bits)/(64 kbps) = 10 segundos en transmitir el archivo. A estos 10 segundos tenemos que sumarles el tiempo de establecimiento del circuito, lo que da como resultado 10,5 segundos de tiempo total de transmisión del archivo. Observe que el tiempo de transmisión es independiente del número de enlaces: el tiempo de transmisión será 10 segundos independientemente de que el circuito terminal a terminal pase a través de un enlace o de cien enlaces. (El retardo real terminal a terminal también incluye un retardo de propagación; véase la Sección 1.4.)

Conmutación de paquetes

Las aplicaciones distribuidas intercambian **mensajes** para llevar a cabo sus tareas. Los mensajes pueden contener cualquier cosa que el diseñador del protocolo desee. Los mensajes pueden realizar una función de control (por ejemplo, los mensajes de saludo "Hola" del ejemplo anterior sobre establecimiento de la comunicación) o pueden contener datos, como por ejemplo un mensaje de correo electrónico, una imagen JPEG o un archivo de audio MP3. En las redes de computadoras modernas, el origen divide los mensajes largos en fragmentos de datos más pequeños que se conocen como **paquetes**. Entre el origen y el destino, cada uno de estos paquetes viaja a través de los enlaces de comunicaciones y de los **conmutadores de paquetes** (de los que existen dos tipos predominantes: los routers y los switches de la capa de enlace). Los paquetes se transmiten a través de cada enlace de comunicaciones a una velocidad igual a la velocidad de transmisión *máxima* del enlace.

La mayoría de los conmutadores de paquetes emplean el método de **transmisión de almacenamiento y reenvío** en las entradas de los enlaces. Transmisión de almacenamiento y reenvío significa que el conmutador tiene que recibir el paquete completo antes de poder comenzar a transmitir el primer bit del paquete al enlace de salida. Por tanto, los conmutadores de paquetes de almacenamiento y reenvío añaden un retardo de almacenamiento y reenvío en la entrada de cada enlace existente a lo largo de la ruta que debe seguir el paquete. Veamos el tiempo que se tarda en enviar un paquete de L bits desde un host a otro host en una red de conmutación de paquetes. Supongamos que existen Q enlaces entre los dos hosts, y que la velocidad en cada uno de ellos es igual a R bps. Suponemos que éste es el único paquete presente en la red. En primer lugar, el paquete tiene que enviarse a través del primer enlace que sale del host A, lo que consume un tiempo de L/R segundos. A continuación, tiene que ser transmitido por cada uno de los $Q - 1$ enlaces restantes; es decir, se tiene que almacenar y reenviar $Q - 1$ veces, añadiéndose cada vez un retardo de almacenamiento y reenvío de L/R. Por tanto, el retardo total es igual a QL/R.

Cada conmutador de paquetes tiene varios enlaces conectados a él y para cada enlace conectado, el conmutador de paquetes dispone de un **buffer de salida** (también denominado **cola de salida**), que almacena los paquetes que el router enviará a través de dicho enlace. El buffer de salida desempeña un papel clave en la conmutación de paquetes. Si un paquete entrante tiene que ser transmitido a través de un enlace, pero se encuentra con que el enlace está ocupado transmitiendo otro paquete, el paquete entrante tendrá que esperar en el buffer de salida. Por tanto, además de los retardos de almacenamiento y reenvío, los paquetes se

ven afectados por los **retardos de cola** del buffer de salida. Estos retardos son variables y dependen del nivel de congestión de la red. Puesto que la cantidad de espacio en el buffer es finita, un paquete entrante puede encontrarse con que el buffer está completamente lleno con otros paquetes que esperan a ser transmitidos. En este caso, se producirá una **pérdida de paquetes**, bien el paquete que acaba de llegar o uno que ya se encuentra en la cola será descartado. Si volvemos a la analogía de los restaurantes vista anteriormente en esta sección, el retardo de cola es análogo a la cantidad de tiempo que usted pierde en el bar del restaurante esperando a que una mesa se quede libre. La pérdida de paquetes es análoga a que el camarero le comunique que es mejor que vaya a otro restaurante porque ya hay demasiadas personas esperando en el bar para conseguir una mesa.

La Figura 1.14 ilustra una red de conmutación de paquetes simple. En esta figura y en las siguientes, los paquetes se han representado mediante bloques tridimensionales. El ancho de un bloque representa el número de bits que contiene el paquete. En esta figura, todos los paquetes tienen el mismo ancho y, por tanto, la misma longitud. Suponga ahora que los hosts A y B están enviando paquetes al host E. En primer lugar, los hosts A y B envían sus paquetes a través de los enlaces Ethernet a 10 Mbps hasta el primer conmutador de paquetes. A continuación, éste dirige los paquetes al enlace de 1,5 Mbps. Si la velocidad de llegada de los paquetes al conmutador excede la velocidad a la que el conmutador puede reenviar los paquetes a través del enlace de salida de 1,5 Mbps, se producirá congestión a medida que los paquetes se pongan en cola en el buffer de salida del enlace antes de poder ser transmitidos. En la Sección 1.4 examinaremos más detalladamente el retardo de cola.

Conmutación de paquetes frente a conmutación de circuitos: multiplexación estadística

Ahora que ya hemos descrito las tecnologías de conmutación de circuitos y de paquetes, vamos a pasar a compararlas. Los detractores de la tecnología de conmutación de paquetes a menudo han argumentado que esta tecnología no es adecuada para los servicios en tiempo real, como por ejemplo las llamadas telefónicas y las videoconferencias, porque sus retardos

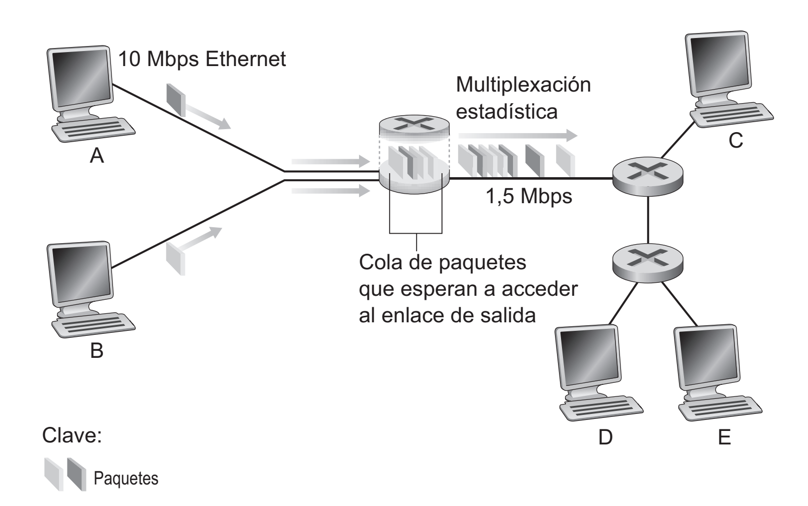


Figura 1.14 • Conmutación de paquetes.

terminal a terminal son variables e impredecibles (a causa principalmente de que los retardos de cola de los paquetes son variables e impredecibles). Por otro lado, los partidarios de la conmutación de paquetes argumentan que (1) ofrece una mejor compartición del ancho de banda que la tecnología de conmutación de circuitos y (2) es más sencilla, más eficiente y menos cara de implementar que la conmutación de circuitos. Puede encontrar una exposición interesante acerca de la conmutación de paquetes frente a la conmutación de circuitos en [Molinero-Fernández 2002]. Generalmente, las personas que no se molestan en reservar mesa en un restaurante prefieren la conmutación de paquetes a la conmutación de circuitos.

¿Por qué es más eficiente la conmutación de paquetes? Veamos un ejemplo sencillo. Suponga que varios usuarios comparten un enlace de 1 Mbps. Suponga también que cada usuario alterna entre periodos de actividad (cuando genera datos a una velocidad constante de 100 kbps) y periodos de inactividad (cuando no genera datos). Además, suponga que un usuario sólo está activo un 10 por ciento del tiempo (y está inactivo tomando café durante el 90 por ciento del tiempo restante). Con la tecnología de conmutación de circuitos, tienen que *reservarse* 100 kbps para *cada* usuario todas las veces. Por ejemplo, en una red de conmutación de circuitos con multiplexación TDM, si un marco de un segundo se divide en 10 particiones de 100 ms, entonces cada usuario tendría asignada una partición por marco.

Por tanto, el enlace de conmutación de circuitos sólo podrá dar soporte a 10 (= 1 Mbps/100 kbps) usuarios simultáneamente. En el caso de utilizar la conmutación de paquetes, la probabilidad de que un determinado usuario esté activo es 0,1 (es decir, del 10 por ciento). Si hay 35 usuarios, la probabilidad de que 11 o más usuarios estén activos simultáneamente es aproximadamente igual a 0,0004. (El Problema P7 indica cómo se obtiene esta probabilidad.) Cuando hay 10 o menos usuarios activos a la vez (lo que ocurre con una probabilidad del 0,9996), la velocidad acumulada de llegada de los datos es menor o igual a 1 Mbps, la velocidad de salida del enlace. Por tanto, cuando el número de usuarios activos es 10 o menor, los paquetes fluyen a través del enlace prácticamente sin retardo, como en el caso de la tecnología de conmutación de circuitos. Cuando hay más de 10 usuarios activos simultáneamente, entonces la velocidad acumulada de llegada de los paquetes excede la capacidad de salida del enlace y la cola de salida comenzará a crecer. (Continúa creciendo hasta que la velocidad acumulada de entrada cae por debajo de 1 Mbps, punto en el que la longitud de la cola comenzará a disminuir.) Puesto que la probabilidad de que haya más de 10 usuarios conectados a la vez es muy baja en este ejemplo, la conmutación de paquetes proporciona prácticamente el mismo rendimiento que la conmutación de circuitos, *pero lo hace permitiendo que haya un número de usuarios más de tres veces superior.*

Consideremos ahora otro ejemplo sencillo. Suponga que hay 10 usuarios y que de repente un usuario genera 1.000 paquetes de 1.000 bits, mientras que los usuarios restantes permanecen inactivos y no generan paquetes. Con la tecnología de conmutación de circuitos con multiplexación TDM con 10 particiones por marco y con cada partición formada por 1.000 bits, el usuario activo sólo puede emplear su partición por marco para transmitir los datos, mientras que las restantes nueve particiones de cada marco permanecen inactivas. Transcurrirán 10 segundos antes de que el millón de bits de datos del usuario activo hayan sido transmitidos. Sin embargo, con la conmutación de paquetes, el usuario activo puede enviar de forma continuada sus paquetes a la velocidad máxima del enlace de 1 Mbps, ya que no hay ningún otro usuario generando paquetes que tengan que ser multiplexados con los paquetes del usuario activo. En este caso, todos los datos del usuario activo se transmitirán en un segundo.

Los ejemplos anteriores han servido para ilustrar dos casos en los que el rendimiento de la tecnología de conmutación de paquetes puede resultar superior a la de la conmutación de circuitos. También ha quedado patente la diferencia crucial entre las dos formas de compartir la velocidad de transmisión del enlace entre varios flujos de datos. La conmutación de circuitos preasigna el uso del enlace de transmisión independientemente de la demanda, con lo que el tiempo de enlace asignado, pero innecesario, se desperdicia. Por el contrario, la conmutación de paquetes asigna el uso del enlace *bajo demanda.* La capacidad de transmisión del enlace se compartirá paquete a paquete sólo entre aquellos usuarios que tienen paquetes que transmitir a través del enlace. La compartición de recursos bajo petición (en lugar de por preasignación) a veces se denomina multiplexación **estadística** de recursos.

Aunque hoy en día las redes de telecomunicaciones predominantes son las de conmutación de circuitos y de paquetes, realmente se está tendiendo hacia las redes de conmutación de paquetes. Incluso muchas de las redes de telefonía de conmutación de circuitos actuales se están migrando lentamente a redes de conmutación de paquetes. En particular, las redes telefónicas suelen emplear la conmutación de paquetes en la parte internacional, que es la más cara de una llamada telefónica.

1.3.2 ¿Cómo atraviesan los paquetes las redes de conmutación de paquetes?

Anteriormente hemos dicho que un router toma un paquete entrante en uno de sus enlaces de comunicaciones, pero ¿cómo el router determina el enlace por el que deberá reenviar el paquete? En realidad, los diferentes tipos de redes pueden hacer esto de diversas formas. En este capítulo de introducción vamos a describir un método popular, el método empleado por Internet.

En Internet, cada paquete que atraviesa la red contiene en su cabecera la dirección del destino del paquete. Al igual que las direcciones postales, esta dirección tiene una estructura jerárquica. Cuando llega un paquete a un router de la red, el router examina una parte de la dirección de destino del paquete y lo reenvía a un router adyacente. Más específicamente, cada router dispone de una **tabla de reenvío** que asigna las direcciones de destino (o una parte de las mismas) a los enlaces salientes. Cuando llega un paquete a un router, éste examina la dirección y busca en su tabla esa dirección de destino para localizar el enlace de salida apropiado. A continuación, el router dirige el paquete a ese enlace de salida.

Acabamos de ver que un router utiliza la dirección de destino de un paquete para indexar una tabla de reenvío y determinar el enlace de salida apropiado. Pero esta afirmación nos lleva a la siguiente pregunta: ¿cómo se definen las tablas de reenvío? ¿Se configuran manualmente en cada router o Internet utiliza un procedimiento más automatizado? Estas cuestiones se abordan en detalle en el Capítulo 4, pero para ir abriendo boca, diremos que Internet dispone de una serie de protocolos de enrutamiento especiales que se utilizan para definir automáticamente las tablas de reenvío. Por ejemplo, un protocolo de enrutamiento determina la ruta más corta desde cada router hasta cada destino y el uso de la ruta más corta da como resultado la configuración de las tablas de reenvío en los routers.

El proceso de enrutamiento de terminal a terminal es análogo al que sigue el conductor de un automóvil que no utiliza un mapa, sino que prefiere preguntar cómo llegar hasta una determinada dirección. Por ejemplo, suponga que Juan sale de Filadelfia y tiene que llegar al 156 de Lakeside Drive en Orlando, Florida. Lo primero que hace Juan es dirigirse a la estación de servicio más próxima y preguntar cómo llegar a su destino. El empleado se

queda con el nombre del estado Florida, y le dice que debe tomar la autopista interestatal I-95 Sur y que existe una entrada a la misma nada más salir de la estación de servicio. También le dice a Juan que una vez que haya entrado en Florida, pregunte a alguien cómo llegar a su destino. Así, Juan toma la I-95 Sur hasta Jacksonville, Florida, lugar donde vuelve a preguntar también en una estación de servicio. El dependiente extrae de la dirección la información que hace referencia a Orlando y le dice que debe continuar por la I-95 hasta Daytona Beach y que luego pregunte. En otra estación de servicio de Daytona Beach, el empleado de nuevo extrae la información referente a Orlando y le dice que tomando la I-4 llegará directamente a Orlando. Juan toma la I-4 y la abandona en la salida que indica a Orlando. De nuevo se detiene en otra gasolinera y esta vez el dependiente extrae la parte de la información de la dirección referente a Lakeside Drive y le indica la carretera que debe seguir para llegar allí. Una vez que Juan se encuentra en Lakeside Drive, pregunta a un niño que va en bicicleta cómo llegar a su destino. El niño extrae el dato 156 de la dirección y le señala una casa. Por fin, Juan ha llegado a su destino.

En esta analogía, los dependientes de las estaciones de servicio y el niño de la bicicleta son los routers, y sus tablas de reenvío, que son sus cerebros, se han ido configurando a lo largo de años de experiencia.

¿Cómo podríamos ver la ruta terminal a terminal que siguen los paquetes en Internet? Le invitamos a que utilice el programa Traceroute, visitando el sitio http://www.traceroute.org. (Para obtener más información acerca de Traceroute, consulte la Sección 1.4.)

1.3.3 Redes troncales de Internet y proveedores ISP

Anteriormente hemos visto que los sistemas terminales (los PC de usuario, las PDA, los servidores web, los servidores de correo electrónico, etc.) se conectan a Internet a través de un ISP local. El ISP puede proporcionar conectividad cableada o inalámbrica, mediante una amplia variedad de tecnologías de acceso, entre las que se incluyen DSL, cable, FTTH, Wi-Fi, celular y WiMAX. Observe que el ISP local no tiene que ser una compañía telefónica ni una compañía de cable: puede ser, por ejemplo, una universidad (que proporciona acceso a Internet a los estudiantes, al personal y a las facultades) o una empresa (que proporciona acceso a sus empleados). Pero la conexión de los usuarios finales y de los proveedores de contenido a los ISP locales es sólo una pequeña parte del problema de conectar los cientos de millones de sistemas terminales y los cientos de miles de redes que conforman Internet. Internet es una *red de redes* y entender esta afirmación es fundamental para resolver este puzzle.

En la red pública Internet, los ISP de acceso situados en la frontera de Internet están conectados al resto de Internet a través de una jerarquía de niveles de proveedores ISP, como se muestra en la Figura 1.15. Los ISP de acceso se sitúan en el nivel inferior de esta jerarquía. En el nivel superior de la jerarquía se encuentran en un número relativamente más pequeño los **ISP de nivel 1**. Por un lado, un ISP de nivel 1 es igual que cualquier red (está formado por enlaces y routers, y está conectado a otras redes). Sin embargo, estos ISP presentan otras características que los hace especiales. La velocidad de enlace suele ser de 622 Mbps o superior, por lo que los ISP de nivel 1 de mayor tamaño disponen de enlaces en el rango comprendido entre 2,5 y 10 Gbps; en consecuencia, sus routers deben poder reenviar los paquetes a velocidades extremadamente altas. Los ISP de nivel 1 también se caracterizan por lo siguiente:

- Están conectados directamente a *cada uno* de los restantes ISP de nivel 1.

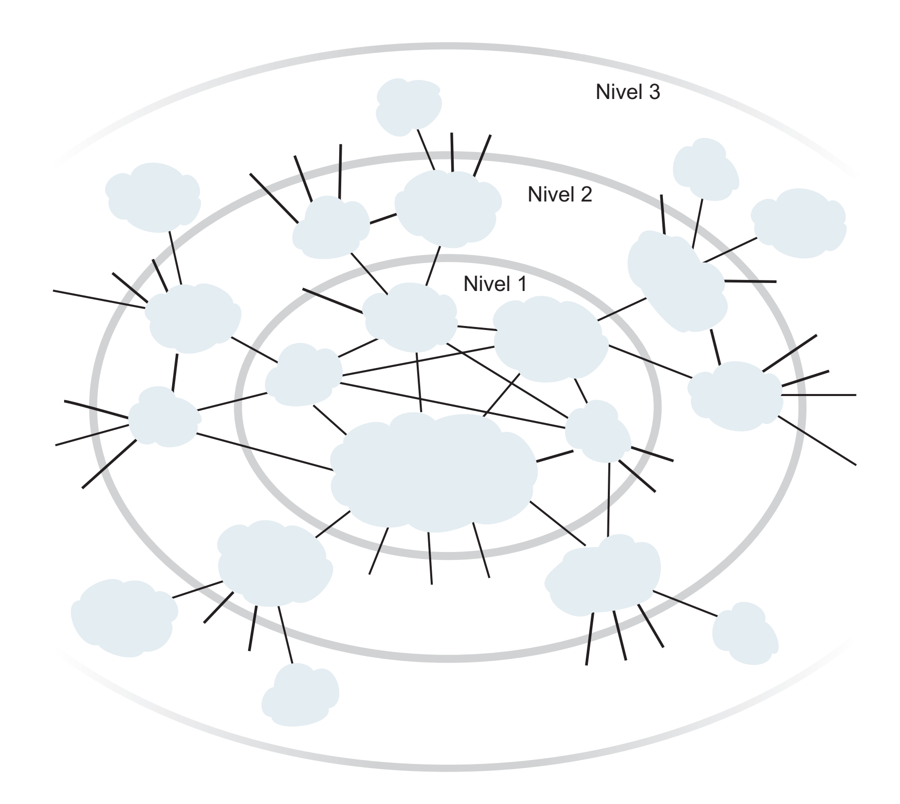


Figura 1.15 • Interconexión de los ISP.

- Están conectados a un gran número de ISP de nivel 2 y a otras redes cliente.
- Proporcionan cobertura internacional.

Los ISP de nivel 1 también se conocen como redes **troncales de Internet**. Entre estos ISP se incluyen Sprint, Verizon, MCI (anteriormente UUNet/WorldCom), AT&T, NTT, Level3, Qwest y Cable & Wireless. Resulta curioso que, oficialmente, no existe ningún grupo que conceda el estatus de nivel 1; como reza el dicho: si tienes que preguntar si eres miembro de un grupo, es que probablemente no lo eres.

Por regla general, un ISP de nivel 2 tiene cobertura regional o nacional y lo que es más importante, sólo está conectado a unos pocos ISP de nivel 1 (véase la Figura 1.15). Por tanto, para llegar a gran parte de la red Internet global, un ISP de nivel 2 tiene que enrutar el tráfico a través de uno de los ISP de nivel 1 a los que está conectado. Se dice que un ISP de nivel 2 es un **cliente** del ISP de nivel 1 al que está conectado y que el ISP de nivel 1 es un **proveedor** de dicho cliente. Muchas instituciones y empresas de gran tamaño conectan sus redes empresariales directamente a un ISP de nivel 1 o de nivel 2, convirtiéndose así en un cliente de dicho ISP. Un ISP proveedor cobra unas determinadas tasas al ISP cliente, que normalmente dependen de la velocidad de transmisión del enlace que los conecta. Una red de nivel 2 también se puede conectar directamente a otras redes de nivel 2, en cuyo caso el tráfico puede fluir entre las dos redes de nivel 2 sin tener que pasar a través de una red de nivel 1. Por debajo de los ISP de nivel 2 se encuentran los ISP de nivel inferior, que se conectan a Internet a través de uno o más ISP de nivel 2. En el nivel más bajo de la jerarquía

se encuentran los ISP de acceso. Para complicar aún más las cosas, algunos proveedores de nivel 1 también son proveedores de nivel 2 (es decir, están integrados verticalmente), que venden directamente acceso a Internet a los usuarios finales y proveedores de contenido, así como a los ISP del nivel inferior. Cuando dos ISP están conectados directamente entre sí en el mismo nivel, se dice que son **igualitarios**. Existe un interesante estudio [Subramanian 2002] que intenta definir la estructura en niveles de Internet de forma más precisa estudiando la topología de Internet en función de las relaciones cliente-proveedor y las relaciones entre iguales. Consulte [Van der Berg 2008] para ver una explicación bastante comprensible acerca de las relaciones entre iguales y cliente-proveedor.

Dentro de la red de un ISP, los puntos en los que el ISP se conecta a otros ISP (sean de nivel inferior, superior o del mismo nivel dentro de la jerarquía) se conocen como **Puntos de presencia (POP, *Point of Presence*)**. Un POP es simplemente un grupo de uno o más routers de la red del ISP en los que los routers de otros ISP o de las redes que pertenecen a los clientes del ISP pueden conectarse. Un proveedor de nivel 1 normalmente tiene muchos POP dispersos por distintas localizaciones geográficas dentro de la red, con múltiples redes y otros ISP conectados a cada POP. Normalmente, cuando una red cliente tiene que conectarse al POP de un proveedor, alquila un enlace de alta velocidad de un proveedor de telecomunicaciones de una tercera empresa y conecta directamente uno de sus routers a un router ubicado en el POP del proveedor. Además, dos ISP pueden disponer de varios puntos de conexión entre iguales, conectándose entre sí en múltiples pares de POP.

En resumen, la topología de Internet es compleja y está formada por docenas de ISP de nivel 1 y de nivel 2 y por miles de ISP de nivel inferior. La cobertura de los ISP puede ser muy variada, pudiéndose extender a varios continentes y océanos hasta estar limitada a pequeñas regiones del mundo. Los ISP del nivel inferior se conectan a los ISP de los niveles superiores y éstos a su vez se interconectan entre sí. Los usuarios y los proveedores de contenido son clientes de los ISP de nivel inferior y éstos son clientes de los ISP de nivel superior.



1.4 Retardos, pérdidas y tasa de transferencia en las redes de conmutación de paquetes

En la Sección 1.1 hemos dicho que Internet puede verse como una infraestructura que proporciona servicios a aplicaciones distribuidas que se ejecutan en sistemas terminales. Idealmente, desearíamos que los servicios de Internet pudieran transportar tantos datos como quisiéramos entre cualesquiera dos sistemas terminales de forma instantánea y sin que tuviera lugar ninguna pérdida de datos. Evidentemente, en la realidad, este objetivo es inalcanzable, ya que necesariamente las redes de computadoras tienen que restringir su tasa de transferencia (la cantidad de datos por segundo que pueden transmitir) entre sistemas terminales, introducir retardos entre los sistemas terminales y perder paquetes. Por una parte, es lamentable que las leyes físicas introduzcan retardos y pérdidas, así como que restrinjan las tasas de transferencia, pero, por otra parte, puesto que las redes de computadoras presentan estos problemas, existen muchas cuestiones interesantes relacionadas con cómo abordarlos, ¡más que suficientes como para llenar un curso sobre redes de computadoras y para motivar cientos de tesis doctorales! En esta sección comenzaremos examinando y cuantificando los retardos, las pérdidas y la tasa de transferencia en las redes de computadoras.

1.4.1 Retardo en las redes de conmutación de paquetes

Recordemos que los paquetes se inician en un host (el origen), atraviesan una serie de routers y terminan su viaje en otro host (el destino). Cuando un paquete viaja de un nodo (host o router) al siguiente nodo (host o router) a lo largo de una ruta, el paquete sufre varios tipos de retardo en *cada* uno de los nodos de dicha ruta. Los más importantes de estos retardos son: el **retardo de procesamiento nodal**, el **retardo de cola**, el **retardo de transmisión** y el **retardo de propagación**; todos estos retardos se suman para proporcionar el **retardo nodal total**. Para adquirir un conocimiento profundo de la tecnología de conmutación de paquetes y de las redes de computadoras, es preciso comprender la naturaleza e importancia de estos retardos.

Tipos de retardos

Utilizaremos la Figura 1.16 para explorar estos retardos. Como parte de la ruta terminal a terminal entre el origen y el destino, un paquete se envía desde el nodo anterior a través del router A hasta el router B. Nuestro objetivo es caracterizar el retardo nodal en el router A. Observe que el router A dispone de un enlace de salida que lleva hasta el router B. Este enlace está precedido por una cola (o buffer). Cuando el paquete llega al router A procedente del nodo anterior, el router A examina la cabecera del paquete para determinar cuál es el enlace de salida apropiado para el paquete y luego dirige dicho paquete a ese enlace. En este ejemplo, el enlace de salida para el paquete es el único que lleva hasta el router B. Un paquete puede transmitirse a través de un enlace sólo si actualmente no se está transmitiendo ningún otro paquete a través de él y si no hay otros paquetes que le precedan en la cola; si el enlace está ocupado actualmente o si existen otros paquetes en la cola esperando para ese enlace, entonces el paquete recién llegado tendrá que ponerse a la cola.

Retardo de procesamiento

El tiempo requerido para examinar la cabecera del paquete y determinar dónde hay que enviarlo es parte del **retardo de procesamiento**. El retardo de procesamiento también incluye otros factores como el tiempo necesario para comprobar los errores de nivel de bit del paquete que se producen al transmitir los bits del paquete desde el nodo anterior al router A. Los retardos de procesamiento en los routers de alta velocidad suelen ser del orden de los microsegundos o menores. Una vez efectuado el procesamiento nodal, el router dirige el paquete a la

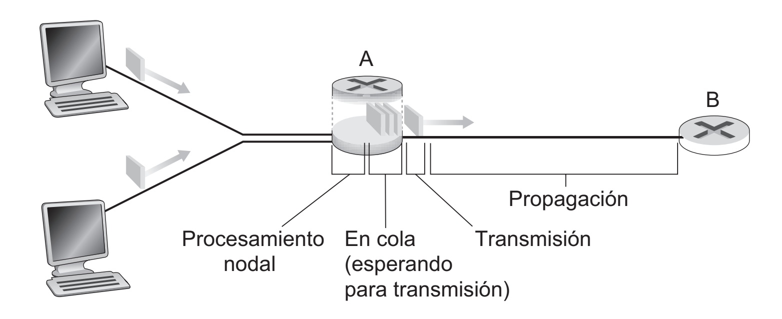


Figura 1.16 • Retardo nodal en el router A.

cola que precede al enlace que lleva al router B. (En el Capítulo 4 estudiaremos los detalles acerca de cómo funciona un router.)

Retardo de cola

En la cola, el paquete experimenta un **retardo de cola** al tener que esperar para ser transmitido a través del enlace. La duración del retardo de cola de un determinado paquete dependerá del número de paquetes que hayan llegado antes a la cola y que están esperando para ser transmitidos por el enlace. Si la cola está vacía y no se está transmitiendo ningún paquete actualmente, entonces el retardo de cola de nuestro paquete será cero. Por el contrario, si hay mucho tráfico y muchos paquetes también están esperando para ser transmitidos, el retardo de cola puede ser grande. Vamos a ver brevemente que el número de paquetes que un paquete entrante puede esperar encontrar es una función de la intensidad y de la naturaleza del tráfico que llega a la cola. En la práctica, los retardos de cola pueden ser del orden de microsegundos a milisegundos.

Retardo de transmisión

Suponiendo que los paquetes se transmiten de manera que el primero que llega es el primero que sale, lo que es una práctica común en las redes de conmutación de paquetes, nuestro paquete sólo puede ser transmitido después de que todos los paquetes que hayan llegado antes que él hayan sido transmitidos. Sea la longitud del paquete igual a L bits y la velocidad de transmisión del enlace del router A hasta el router B igual a R bits/segundo. Entonces, por ejemplo, para un enlace Ethernet a 10 Mbps, la velocidad es R = 10 Mbps; para un enlace Ethernet a 100 Mbps, la velocidad será R = 100 Mbps. El **retardo de transmisión** (también denominado retardo de almacenamiento y reenvío, como hemos visto en la Sección 1.3) será igual a L/R. Este es el tiempo necesario para introducir (es decir, transmitir) todos los bits del paquete por el enlace. Normalmente, en la práctica, los retardos de transmisión son del orden de los microsegundos a los milisegundos.

Retardo de propagación

Una vez que un bit ha entrado en el enlace, tiene que propagarse hasta el router B. El tiempo necesario para propagarse desde el principio del enlace hasta el router B es el **retardo de propagación**. El bit se propaga a la velocidad de propagación del enlace. Esta velocidad depende del medio físico del enlace (es decir, que el medio sea cable de fibra óptica, cable de cobre de par trenzado, etc.) y está comprendido en el rango entre

$$2 \;\; 10^8 \text{ metros/segundo y } 3 \;\; 10^8 \text{ metros/segundo}$$

que es igual o menor que la velocidad de la luz. El retardo de propagación es igual a la distancia entre dos routers dividida entre la velocidad de propagación. Es decir, el retardo de propagación es igual a d/s, donde d es la distancia entre el router A y el router B y s es la velocidad de propagación del enlace. Una vez que el último bit del paquete se ha propagado hasta el nodo B, éste y todos los bits anteriores del paquete se almacenan en el router B. A continuación, el router B lleva a cabo el reenvío. En las redes de área extensa, los retardos de propagación son del orden de los milisegundos.

Comparación de los retardos de transmisión y de propagación

Los recién llegados al campo de las redes de computadoras en ocasiones tienen dificultades para comprender la diferencia entre el retardo de transmisión y el de propagación. Esta diferencia es sutil pero importante. El retardo de transmisión es la cantidad de tiempo necesario para que el router saque fuera el paquete; es una función de la longitud del paquete y de la velocidad de transmisión del enlace, pero no tiene nada que ver con la distancia existente entre los dos routers. Por el contrario, el retardo de propagación es el tiempo que tarda un bit en propagarse de un router al siguiente; es una función de la distancia entre los dos routers, pero no tiene nada que ver con la longitud del paquete ni con la velocidad de transmisión del enlace.

Veamos una analogía que nos va a permitir clarificar los conceptos de retardo de transmisión y de retardo de propagación. Imagine una autopista en la que hay un puesto de peaje cada 100 kilómetros, como se muestra en la Figura 1.17. Podemos imaginar que los segmentos de autopista entre peajes son los enlaces y las casetas de peaje son los routers. Suponga que los automóviles viajan (es decir, se propagan) por la autopista a una velocidad de 100 km/hora (es decir, cuando un coche sale de un peaje, instantáneamente acelera hasta adquirir la velocidad de 100 km/hora y la mantiene entre puestos de peaje). Supongamos ahora que hay 10 coches que viajan en caravana unos detrás de otros en un orden fijo. Podemos pensar que cada coche es un bit y que la caravana es un paquete. Supongamos también que cada puesto de peaje da servicio (es decir, transmite) a los coches a una velocidad de un coche cada 12 segundos y que es tarde por la noche, por lo que en la autopista sólo se encuentra nuestra caravana de coches. Por último, supongamos que cuando el primer coche de la caravana llega a un peaje, espera en la entrada hasta que los otros nueve coches han llegado y se han detenido detrás de él (así, la caravana completa tiene que almacenarse en el peaje antes de poder ser reenviada). El tiempo necesario para que el peaje ponga a la caravana completa en la autopista es igual a (10 coches)/(5 coches/minuto) = 2 minutos. Este tiempo es análogo al retardo de transmisión de un router. El tiempo necesario para que un coche se desplace desde la salida del peaje hasta el siguiente puesto de peaje es 100 km/(100 km/hora) = 1 hora. Este tiempo es análogo al retardo de propagación. Por tanto, el tiempo que transcurre desde que la caravana queda colocada delante de un peaje hasta que vuelve a quedar colocada delante del siguiente peaje es la suma del tiempo de transmisión y el tiempo de propagación (en este caso, dicho tiempo será igual a 62 minutos).

Profundicemos un poco más en esta analogía. ¿Qué ocurriría si el tiempo de servicio del puesto de peaje invertido en una caravana fuera mayor que el tiempo que tarda un coche en viajar de un peaje al siguiente? Por ejemplo, supongamos que los coches viajan a una velocidad de 1.000 km/hora y que los peajes operan a una velocidad de un coche por minuto. Luego el retardo de desplazarse entre dos puestos de peaje será de 6 minutos y el tiempo que tarda el puesto de peaje en dar servicio a una caravana es de 10 minutos. En este caso, los

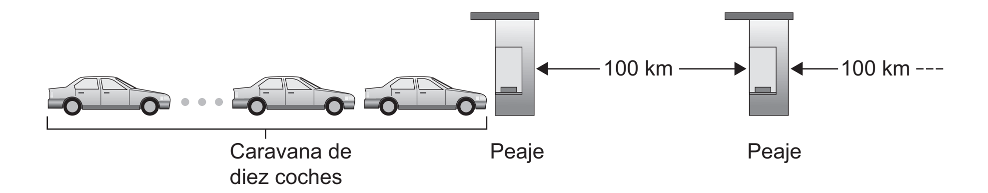


Figura 1.17 • Analogía de la caravana.

primeros coches de la caravana llegarán al segundo puesto de peaje antes de que los últimos coches de la caravana hayan salido del primer peaje. Esta situación también se produce en las redes de conmutación de paquetes: los primeros bits de un paquete pueden llegar a un router mientras que gran parte de los bits restantes del paquete todavía están esperando a ser transmitidos por el router anterior.

Si una imagen vale más que mil palabras, entonces una animación vale más que un millón de palabras. En el sitio web de este libro de texto se proporciona un applet Java interactivo que ilustra y compara los retardos de transmisión y de propagación. Animamos a los lectores a visitar este applet.

Sean d_{proc}, d_{cola}, d_{trans} y d_{prop} los retardos de procesamiento, de cola, de transmisión y de propagación, respectivamente. Entonces el retardo total nodal estará dado por:

$$d_{nodal} = d_{proc} + d_{cola} + d_{trans} + d_{prop}$$

Las contribuciones de estos componentes de retardo pueden variar significativamente. Por ejemplo, d_{prop} puede ser despreciable (digamos un par de microsegundos) para un enlace que conecte dos routers del mismo campus universitario; sin embargo, d_{prop} será igual a cientos de milisegundos para dos routers interconectados mediante un enlace vía satélite geoestacionario y puede ser el término dominante en la expresión que proporciona el retardo d_{nodal}. Del mismo modo, d_{trans} puede ser despreciable o significativo. Su contribución normalmente es despreciable para velocidades de transmisión de 10 Mbps y superiores (por ejemplo, para las redes LAN); sin embargo, puede ser igual a cientos de milisegundos para paquetes grandes de Internet enviados a través del modems de acceso telefónico de baja velocidad. El retardo de procesamiento, d_{proc}, suele ser despreciable; sin embargo, tiene una gran influencia sobre la tasa de transferencia máxima del router, que es la velocidad máxima a la que un router puede reenviar los paquetes.

1.4.2 Retardo de cola y pérdida de paquetes

El componente más complejo e interesante del retardo nodal es el retardo de cola, d_{cola}. De hecho, el retardo de cola es tan importante e interesante en las redes de computadoras que se han escrito miles de artículos y muchos libros sobre él [Bertsekas 1991; Daigle 1991; Kleinrock 1975, 1976; Ross 1995]. Aquí sólo vamos a abordarlo de forma intuitiva y panorámica; los lectores más curiosos pueden echar un vistazo a algunos de los libros que se ocupan de este tema (¡o incluso pueden escribir una tesis doctoral sobre el asunto!). A diferencia de los otros tres retardos (d_{proc}, d_{trans} y d_{prop}), el retardo de cola puede variar de un paquete a otro. Por ejemplo, si llegan 10 paquetes a una cola vacía al mismo tiempo, el primer paquete transmitido no sufrirá retardo de cola, mientras que el último paquete transmitido sufrirá un retardo de cola relativamente largo (mientras espera a que los restantes nueve paquetes sean transmitidos). Por tanto, al caracterizar el retardo de cola, suelen emplearse medidas estadísticas, como el retardo medio de cola, la varianza del retardo de cola y la probabilidad de que el retardo de cola exceda un cierto valor especificado.

¿En qué casos el retardo de cola es grande y en qué casos es insignificante? La respuesta a esta pregunta depende de la velocidad a la que llega el tráfico a la cola, de la velocidad de transmisión del enlace y de la naturaleza del tráfico entrante, es decir, si el tráfico llega periódicamente o a ráfagas. Vamos a profundizar en este punto. Sea a la velocidad media a la que llegan los paquetes a la cola (a se expresa en paquetes/segundo). Recuerde que R es la veloci-

dad de transmisión; es decir, es la velocidad (en bits/segundo) a la que los bits salen de la cola. Con el fin de simplificar, supongamos también que todos los paquetes constan de L bits. Luego la velocidad media a la que llegan los bits a la cola es igual a La bits/segundo. Supongamos por último que la cola es muy grande, por lo que podemos decir que puede almacenar un número infinito de bits. La relación La/R, denominada **intensidad de tráfico**, suele desempeñar un papel importante a la hora de estimar la magnitud del retardo de cola. Si $La/R > 1$, entonces la velocidad media a la que los bits llegan a la cola excede la velocidad a la que los bits pueden ser transmitidos desde la cola. En esta desafortunada situación, la cola tenderá a aumentar sin límite y el retardo de cola se aproximará a ¡infinito! Por tanto, una de las reglas de oro en la ingeniería de tráfico es: *diseñe su sistema de modo que la intensidad de tráfico no sea mayor que 1.*

Veamos ahora el caso en que La/R 1. Aquí la naturaleza del tráfico entrante influye sobre el retardo de cola. Por ejemplo, si los paquetes llegan periódicamente, es decir, llega un paquete cada L/R segundos, entonces todos los paquetes llegarán a una cola vacía y no habrá retardo de cola. Por el contrario, si los paquetes llegan a ráfagas pero de forma periódica, puede aparecer un retardo medio de cola significativo. Por ejemplo, supongamos que llegan simultáneamente N paquetes cada $(L/R)N$ segundos. En este caso, el primer paquete transmitido no tiene asociado un retardo de cola, el segundo paquete transmitido presentará un retardo de cola de L/R segundos y, de forma más general, el n-ésimo paquete transmitido presentará un retardo de cola de $(n - 1)L/R$ segundos. Dejamos como ejercicio para el lector el cálculo del retardo medio de cola de este ejemplo.

Los dos ejemplos de llegada periódica de los paquetes que acabamos de describir se corresponden con casos teóricos. Normalmente, el proceso de llegada a una cola es *aleatorio;* es decir, las llegadas no siguen ningún patrón y los paquetes quedan separados por periodos de tiempo aleatorios. En este caso más realista, la cantidad La/R normalmente no es suficiente para caracterizar completamente las estadísticas del retardo de cola. Aun así, resulta útil tener una idea intuitiva de la magnitud del retardo de cola. En particular, si la intensidad de tráfico es próxima a cero, entonces las llegadas de paquetes serán pocas y estarán bastante espaciadas entre sí, por lo que será improbable que un paquete que llegue a la cola se encuentre con que hay otro paquete en la cola. Por tanto, el retardo medio de cola será próximo a cero. Por el contrario, cuando la intensidad de tráfico es próxima a 1, habrá intervalos de tiempo en los que la velocidad de llegada exceda a la capacidad de transmisión (a causa de las variaciones en la velocidad de llegada de los paquetes), por lo que se formará una cola durante estos periodos de tiempo; si la velocidad de llegada es menor que la capacidad de transmisión, la longitud de la cola disminuirá. Sin embargo, cuando la intensidad de tráfico se aproxime a 1, la longitud media de la cola será cada vez mayor. La dependencia cualitativa del retardo medio de cola con relación a la intensidad de tráfico se muestra en la Figura 1.18.

Un aspecto importante de la Figura 1.18 es el hecho de que cuando la intensidad de tráfico se aproxima a 1, el retardo medio de cola aumenta rápidamente. Un pequeño porcentaje de aumento en la intensidad dará lugar a un incremento en porcentaje muy grande del retardo. Es posible que haya experimentado este fenómeno en una autopista. Si viaja regularmente por una autopista que habitualmente está congestionada, quiere decir que la intensidad de tráfico en esa autopista es próxima a 1. En el caso de que se produzca un suceso que dé lugar a una cantidad de tráfico ligeramente mayor que la usual, los retardos que se experimenten pueden llegar a ser enormes.

Con el fin de que entienda bien lo que son los retardos de cola animamos de nuevo al lector a visitar el sitio web dedicado a este libro, donde se proporciona un applet de Java

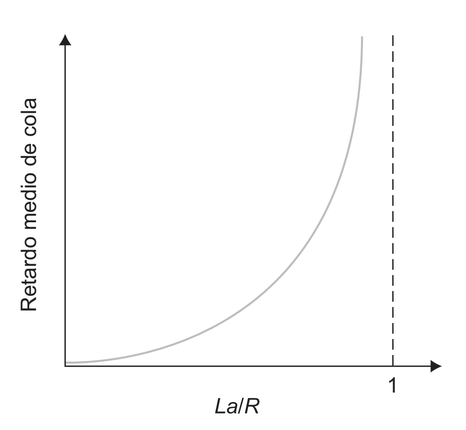


Figura 1.18 • Dependencia del retardo medio de cola con relación a la intensidad de tráfico.

interactivo para una cola. Si establece una velocidad de llegada de los paquetes lo suficientemente alta como para que la intensidad de tráfico sea mayor que 1, comprobará que la cola aumenta lentamente con el tiempo.

Pérdida de paquetes

En la sección anterior hemos supuesto que la cola es capaz de almacenar un número infinito de paquetes. En la práctica, una cola para acceder a un enlace tiene una capacidad finita, aunque la capacidad de la cola depende fundamentalmente del diseño y del coste del router. Puesto que la capacidad de cola es finita, los retardos de los paquetes realmente no se aproximan a infinito cuando la intensidad de tráfico se aproxima a 1. En su lugar, un paquete puede llegar y encontrarse con que la cola está llena. Si no hay sitio para almacenar un paquete, el router lo **elimina**; es decir, el paquete se **pierde**. Este desbordamiento de una cola puede verse también en el applet de Java, cuando la intensidad de tráfico es mayor que 1.

Desde el punto de vista de un sistema terminal, un paquete perdido es un paquete que ha sido transmitido al núcleo de la red pero que nunca sale de la red en su destino. El número de paquetes perdidos aumenta cuando la intensidad de tráfico aumenta. Por tanto, el rendimiento de un nodo suele medirse no sólo en función de los retardos, sino también en función de la probabilidad de pérdida de paquetes. Como veremos en los siguientes capítulos, un paquete perdido puede retransmitirse de terminal a terminal para garantizar que todos los datos sean transferidos desde el origen hasta su destino.

1.4.3 Retardo terminal a terminal

Hasta el momento nos hemos centrado en el retardo nodal, es decir, el retardo en un único router. Ahora vamos a ocuparnos del retardo total entre el origen y el destino. Para entender este concepto, suponga que hay $N - 1$ routers entre el host de origen y el host de destino. Suponga también, por el momento, que la red no está congestionada (por lo que los retardos de cola son despreciables), el retardo de procesamiento en cada router y en el host de origen es d_{proc}, la velocidad de transmisión de salida de cada router y del host de origen es de

R bits/segundo y el retardo de propagación en cada enlace es igual a d_{prop}. Los retardos nodales se suman para proporcionar el retardo terminal a terminal, luego

$$d_{terminal\text{-}terminal} = N\,(d_{proc} + d_{trans} + d_{prop})$$

donde, de nuevo, $d_{trans} = L/R$, siendo *L* el tamaño del paquete. Dejamos para el lector el ejercicio de generalizar esta fórmula para el caso en que los retardos en los nodos sean diferentes y exista un retardo medio de cola en cada nodo.

Traceroute

Para ver el orden de magnitud del retardo terminal a terminal de una red de computadoras, podemos utilizar el programa Traceroute. Se trata de un programa simple que se puede ejecutar en cualquier host de Internet. Cuando el usuario especifica un nombre de host de destino, el programa del host de origen envía varios paquetes especiales al destino. A medida que estos paquetes se dirigen a su destino, pasan a través de una serie de routers. Cuando un router recibe uno de estos paquetes especiales, devuelve al origen un mensaje corto que contiene el nombre y la dirección del router.

Más concretamente, suponga que hay $N - 1$ routers entre el origen y el destino. Luego el origen enviará *N* paquetes especiales a la red dirigidos al destino final. Estos *N* paquetes especiales se marcan de *1* a *N,* quedando el primer paquete marcado como *1* y el último como *N*. Cuando el router *n*-ésimo recibe el paquete *n*-ésimo marcado como *N,* el router no reenvía el paquete hacia su destino, sino que devuelve un mensaje al origen. Cuando el host de destino recibe el paquete *n*-ésimo, también devuelve un mensaje al origen. El origen registra el tiempo transcurrido entre el momento en que envió un paquete y el momento en que recibe el correspondiente mensaje de vuelta; también registra el nombre y la dirección del router (o del host de destino) que devuelve el mensaje. De esta forma, el origen puede reconstruir la ruta seguida por los paquetes que fluyen desde el origen hasta el destino, y el origen puede determinar los retardos de ida y vuelta de todos los routers que intervienen en el proceso. Traceroute repite el proceso que acabamos de describir tres veces, de modo que el origen realmente envía *3* • *N* paquetes al destino. El documento RFC 1393 describe en detalle el programa Traceroute.

He aquí un ejemplo de la salida proporcionada por el programa Traceroute, en el que se ha trazado la ruta desde el host de origen gaia.cs.umass.edu (en la Universidad de Massachusetts) al host cis.poly.edu (en la Universidad Politécnica de Brooklyn). La salida consta de seis columnas: la primera de ellas contiene el valor *n* descrito anteriormente, es decir, el número del router a lo largo de la ruta; la segunda columna especifica el nombre del router; la tercera indica la dirección del router (con el formato xxx.xxx.xxx.xxx); las tres últimas columnas especifican los retardos de ida y vuelta correspondientes a los tres experimentos. Si el origen recibe menos de tres mensajes de cualquier router (debido a la pérdida de paquetes en la red), Traceroute incluye un asterisco justo después del número de router y proporciona menos de tres tiempos de ida y vuelta para dicho router.

```
1 cs-gw (128.119.240.254) 1.009 ms 0.899 ms 0.993 ms
2 128.119.3.154 (128.119.3.154) 0.931 ms 0.441 ms 0.651 ms
3 border4-rt-gi-1-3.gw.umass.edu (128.119.2.194) 1.032 ms 0.484 ms 0.451 ms
4 acr1-ge-2-1-0.Boston.cw.net (208.172.51.129) 10.006 ms 8.150 ms 8.460 ms
5 agr4-loopback.NewYork.cw.net (206.24.194.104) 12.272 ms 14.344 ms 13.267 ms
```

```
6 acr2-loopback.NewYork.cw.net (206.24.194.62) 13.225 ms 12.292 ms 12.148 ms
7 pos10-2.core2.NewYork1.Level3.net (209.244.160.133) 12.218 ms 11.823 ms 11.793 ms
8 gige9-1-52.hsipaccess1.NewYork1.Level3.net (64.159.17.39) 13.081 ms 11.556 ms 13.297 ms
9 p0-0.polyu.bbnplanet.net (4.25.109.122) 12.716 ms 13.052 ms 12.786 ms
10 cis.poly.edu (128.238.32.126) 14.080 ms 13.035 ms 12.802 ms
```

Podemos ver en esta traza que existen nueve routers entre el origen y el destino. La mayor parte de estos routers tiene un nombre y todos ellos tienen direcciones. Por ejemplo, el nombre del router 3 es `border4-rt-gi-1-3.gw.umass.edu` y su dirección es `128.119.2.194`. Si observamos los datos proporcionados para este mismo router, vemos que en la primera de las tres pruebas el retardo de ida y vuelta entre el origen y el router ha sido de 1,03 milisegundos. Los retardos de ida y vuelta para las dos pruebas siguientes han sido 0,48 y 0,45 milisegundos, respectivamente. Estos retardos de ida y vuelta contienen todos los retardos que acabamos de estudiar, incluyendo los retardos de transmisión, de propagación, de procesamiento del router y de cola. Puesto que el retardo de cola varía con el tiempo, el retardo de ida y vuelta del paquete *n* envíado al router *n* puede, en ocasiones, ser mayor que el retardo de ida y vuelta del paquete *n*+1 enviado al router *n*+1. Efectivamente, puede observar este fenómeno en el ejemplo anterior: los retardos correspondientes al router 6 son mayores que los correspondientes al router 7.

¿Desea probar el programa Traceroute? Le recomendamos *vivamente* que visite el sitio http:// www.traceroute.org, donde se proporciona una interfaz web a una extensa lista de orígenes para el trazado de rutas. Seleccione un origen y especifique el nombre de host de cualquier destino. El programa Traceroute hará entonces todo el trabajo. Hay disponibles diversos programas software gratuitos que proporcionan una interfaz gráfica para Traceroute; uno de nuestros programa favoritos es PingPlotter [PingPlotter 2009].

Retardos de los sistemas terminales, de las aplicaciones y otros

Además de los retardos de procesamiento, de transmisión y de propagación, en los sistemas terminales pueden existir retardos adicionales significativos. Por ejemplo, los modems de acceso telefónico introducen un retardo de modulación/codificación, que puede ser del orden de decenas de milisegundos (los retardos de modulación/codificación para otras tecnologías de acceso, como Ethernet, modems por cable y DSL, son menos significativos y suelen ser despreciables). Un sistema terminal que desea transmitir un paquete a través de un medio compartido (por ejemplo, en un escenario WiFi o Ethernet) puede retardar su transmisión *a propósito* como parte de su protocolo, para compartir el medio con otros sistemas terminales. Veremos estos protocolos en detalle en el Capítulo 5. Otro retardo importante es el retardo de empaquetamiento del medio, que aparece en las aplicaciones de Voz sobre IP (VoIP, *Voice-over-IP*). En VoIP, el lado emisor debe, en primer lugar, rellenar un paquete con voz digitalizada codificada antes de pasar el paquete a Internet. El tiempo que se tarda en rellenar un paquete (lo que se denomina retardo de empaquetamiento) puede ser significativo y puede repercutir en la calidad percibida por el usuario de una llamada VoIP. Este problema se abordará más detalladamente en uno de los problemas del final del capítulo.

1.4.4 Tasa de transferencia en las redes de computadoras

Además de los retardos y la pérdida de paquetes, otra medida crítica de rendimiento de las redes de computadoras es la tasa de transferencia de terminal a terminal. Para definir la tasa

de transferencia, consideremos la transferencia de un archivo de gran tamaño desde el host A al host B a través de una red. Por ejemplo, esta transferencia podría consistir en transferir un clip de vídeo de gran tamaño desde un par (*peer*) a otro en un sistema de compartición de archivos P2P. La **tasa de transferencia instantánea** en cualquier instante de tiempo es la velocidad (en bits/segundo) a la que el host B recibe el archivo. (Muchas aplicaciones, incluyendo muchos sistemas de compartición de archivos P2P, muestran la tasa de transferencia instantánea durante las descargas en la interfaz del usuario; ¡es posible que ya se haya fijado anteriormente en este detalle!) Si el archivo consta de F bits y la transferencia dura T segundos hasta que el host B recibe los F bits, entonces la **tasa media de transferencia** del archivo es igual a F/T bits/segundo. En algunas aplicaciones, tales como la telefonía por Internet, es deseable tener un retardo pequeño y una tasa de transferencia instantánea por encima de un cierto umbral (por ejemplo, por encima de 24 kbps para ciertas aplicaciones de telefonía por Internet y por encima de 256 kbps para las aplicaciones de vídeo en tiempo real). Para otras aplicaciones, entre las que se incluyen aquéllas que implican la transferencia de archivos, el retardo no es crítico, pero es deseable que la tasa de transferencia sea lo más alta posible.

Con el fin de comprender mejor el importante concepto de tasa de transferencia, vamos a ver algunos ejemplos. La Figura 1.19(a) muestra dos sistemas terminales, un servidor y un cliente, conectados mediante dos enlaces de comunicaciones y un router. Consideremos la tasa de transferencia para transmitir un archivo desde el servidor al cliente. Sea R_s la velocidad del enlace entre el servidor y el router, y sea R_c la velocidad del enlace entre el router y el cliente. Supongamos que los únicos bits que están siendo enviados a través de la red son los que van desde el servidor al cliente. En este escenario ideal, ¿cuál es la tasa de transferencia del servidor al cliente? Para responder a esta pregunta, podemos pensar en los bits como en un *fluido* y en los enlaces de comunicaciones como en las *tuberías*. Evidentemente, el servidor no puede bombear los bits a través de su enlace a una velocidad mayor que R_s bps; y el router no puede reenviar los bits a una velocidad mayor que R_c bps. Si $R_s < R_c$, entonces los bits bombeados por el servidor "fluirán" a través del router y llegarán al cliente a una velocidad de R_s bps, obteniéndose una tasa de transferencia de R_s bps. Si, por el contrario, $R_c < R_s$, entonces el router no podrá reenviar los bits tan rápidamente como los recibe. En

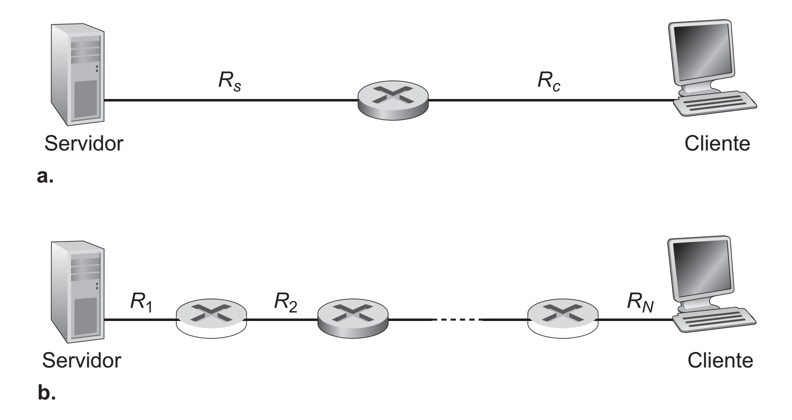


Figura 1.19 • Tasa de transferencia para la transmisión de un archivo desde un servidor a un cliente.

este caso, los bits abandonarán el router a una velocidad de sólo R_c, dando lugar a una tasa de transferencia de terminal a terminal igual a R_c. (Observe también que si continúan llegando bits al router a una velocidad R_s, y siguen abandonando el router a una velocidad igual a R_c, la cantidad de bits en el router que están esperando a ser transmitidos hacia el cliente aumentará constantemente, lo que es una situación nada deseable.) Por tanto, en esta sencilla red de dos enlaces, la tasa de transferencia es mín$\{R_c, R_s\}$, es decir, la velocidad de transmisión del **enlace cuello de botella**. Una vez determinada la tasa de transferencia, podemos aproximar el tiempo que se tardará en transferir un archivo de gran tamaño de F bits desde el servidor al cliente como F/mín$\{R_s, R_c\}$. Veamos un ejemplo concreto. Suponga que está descargando un archivo MP3 de $F = 32$ millones de bits, el servidor tiene una velocidad de transmisión de $R_s = 2$ Mbps y la velocidad del enlace es $R_c = 1$ Mbps. El tiempo necesario para transferir el archivo será igual a 32 segundos. Por supuesto, estas expresiones para la tasa de transferencia y el tiempo de transferencia son únicamente aproximaciones, ya que no se han tenido en cuenta las cuestiones relativas al protocolo y a nivel de paquete.

La Figura 1.19(b) muestra una red con N enlaces entre el servidor y el cliente, siendo las velocidades de transmisión de los N enlaces iguales a $R_1, R_2,..., R_N$. Aplicando el mismo análisis que para la red de dos enlaces, podemos determinar que la tasa de transferencia para transferir un archivo desde el servidor al cliente es mín$\{R_1, R_2,...,R_N\}$, que es de nuevo la velocidad de transmisión del enlace cuello de botella existente en la ruta entre el servidor y el cliente.

Veamos ahora otro ejemplo inspirado en la red Internet de hoy día. La Figura 1.20(a) muestra dos sistemas terminales, un servidor y un cliente, conectados a una red de computadoras. Considere la tasa de transferencia para transmitir un archivo desde el servidor al cliente. El servidor está conectado a la red mediante un enlace de acceso cuya velocidad es R_s y el cliente está conectado a la red mediante un enlace de acceso de velocidad R_c. Supongamos ahora que todos los enlaces existentes en el núcleo de la red de comunicaciones tienen velocidades de transmisión muy altas, muy por encima de R_s y R_c. Ciertamente, hoy en día, el núcleo de Internet está sobredimensionado, con enlaces de alta velocidad que experimentan una congestión muy baja [Akella 2003]. Supongamos también que únicamente se están enviando a la red los bits que se transfieren desde el servidor al cliente. Dado que el núcleo de la red es como una tubería ancha en este ejemplo, la velocidad a la que los bits pueden fluir desde el origen hasta el destino de nuevo es el mínimo de R_s y R_c, es decir, la tasa de transferencia es igual a mín$\{R_s, R_c\}$. Por tanto, normalmente, el factor de restricción de la tasa de transferencia en Internet actualmente es la red de acceso.

Veamos un último ejemplo. Vamos a utilizar la Figura 1.20(b); en ella vemos que hay 10 servidores y 10 clientes conectados al núcleo de la red de computadoras. En este ejemplo, tienen lugar 10 descargas simultáneas, lo que implica a 10 pares cliente-servidor. Supongamos que estas 10 descargas son el único tráfico existente en la red a un mismo tiempo. Como se muestra en la figura, hay un enlace en el núcleo que es atravesado por las 10 descargas. Sea R la velocidad de transmisión de este enlace R. Supongamos que los enlaces de acceso de todos los servidores tienen la misma velocidad R_s, los enlaces de acceso de todos los clientes tienen la misma velocidad R_c y las velocidades de transmisión de todos los enlaces del núcleo (excepto el enlace común de velocidad R) tienen velocidades mucho mayores que R_s, R_c y R. Ahora deseamos saber cuáles son las tasas de transferencia de las descargas. Evidentemente, si la velocidad del enlace común, R, es por ejemplo cien veces mayor que R_s y R_c, entonces la tasa de transferencia de cada descarga será de nuevo mín$\{R_s, R_c\}$. Pero, ¿qué ocurre si la velocidad del enlace común es del mismo orden que R_s

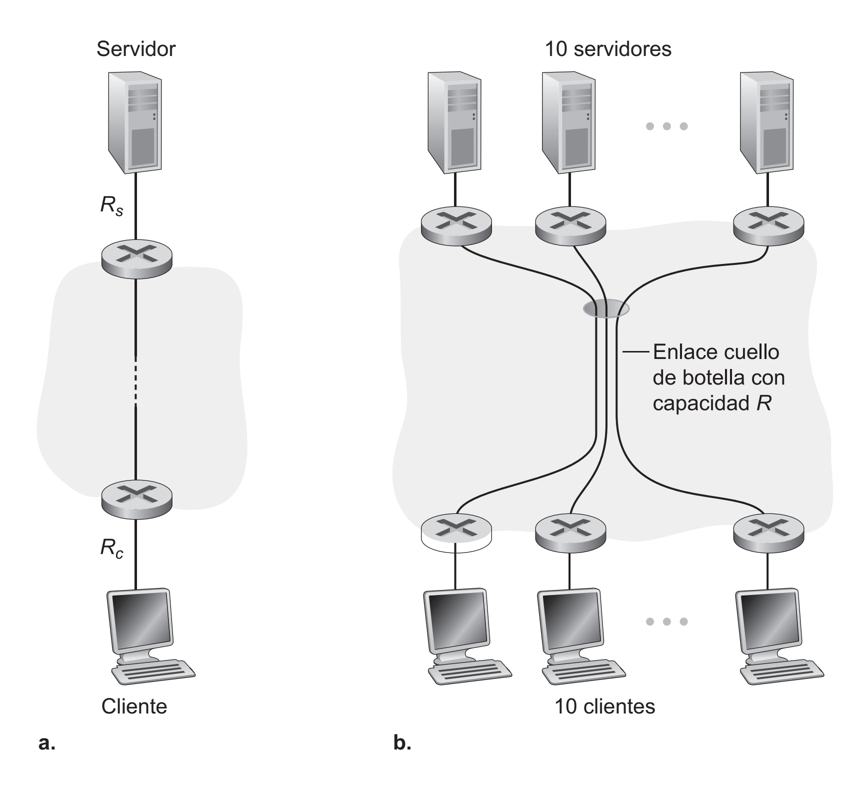


Figura 1.20 • Tasa de transferencia terminal a terminal: (a) un cliente descarga un archivo de un servidor; (b) 10 clientes descargando con 10 servidores.

y R_c? ¿Cuál será la tasa de transferencia en este caso? Veamos un ejemplo concreto. Supongamos que R_s = 2 Mbps, R_c = 1 Mbps, R = 5 Mbps y que el enlace común divide su velocidad de transmisión en partes iguales entre las 10 descargas. Luego, ahora, el cuello de botella para cada descarga ya no se encuentra en la red de acceso, sino en el enlace compartido del núcleo que sólo proporciona una tasa de transferencia de 500 kbps a cada descarga. Luego la tasa de transferencia terminal a terminal para cada descarga ahora se ha reducido a 500 kbps.

Los ejemplos de las Figuras 1.19 y 1.20(a) demuestran que la tasa de transferencia depende de las velocidades de transmisión de los enlaces a través de los que fluyen los datos. Hemos visto que cuando no existe ningún otro tráfico, la tasa de transferencia puede simplemente aproximarse a la velocidad mínima de transmisión a lo largo de la ruta entre el origen y el destino. El ejemplo de la Figura 1.20(b) muestra que, generalmente, la tasa de transferencia depende no sólo de las velocidades de transmisión de los enlaces a lo largo de la ruta, sino también del tráfico existente. En particular, un enlace con una velocidad de transmisión alta puede ser el enlace cuello de botella para la transferencia de un archivo si hay muchos otros flujos de datos atravesando también ese enlace. Examinaremos más detalladamente la tasa de transferencia en las redes de computadoras en los problemas que el lector realizará en su casa y en los capítulos siguientes.

TP2

1.5 Capas de protocolos y sus modelos de servicio

Después de lo que hemos visto hasta aquí, parece que Internet es un sistema *extremadamente* complicado. Hemos visto que son muchas las piezas que conforman Internet: numerosas aplicaciones y protocolos, distintos tipos de sistemas terminales, dispositivos de conmutación de paquetes y diversos tipos de medios para los enlaces. Dada esta enorme complejidad, ¿tenemos alguna esperanza de poder organizar una arquitectura de red o al menos nuestra exposición sobre la misma? Afortunadamente, la respuesta a ambas preguntas es sí.

1.5.1 Arquitectura en capas

Antes de intentar organizar nuestras ideas sobre la arquitectura de Internet, vamos a ver una analogía humana. Realmente, de forma continua estamos tratando con sistemas complejos en nuestra vida cotidiana. Imagine que alguien le pide que describa, por ejemplo, cómo funciona el sistema de líneas aéreas. ¿Qué estructura utilizaría para describir este complejo sistema que emplea agencias de viajes para la venta de billetes, personal para el control de equipajes, puertas de embarque, pilotos, aviones, control de tráfico aéreo y un sistema de ámbito mundial para dirigir los aviones? Una forma de describir este sistema podría consistir en describir la serie de acciones que usted realiza (o que otros realizan para usted) cuando se vuela en un avión. En primer lugar, usted compra un billete, luego factura el equipaje, se dirige a la puerta de embarque y por último sube al avión. El avión despega y se dirige a su destino. Una vez que el avión ha tomado tierra, usted desembarca y recoge su equipaje. Si el viaje ha sido malo, se quejará de ello a la agencia de viajes (lo que, por supuesto, no le servirá de nada). Este escenario se muestra en la Figura 1.21.

Podemos ver algunas analogías con las redes de computadoras: la compañía área le traslada desde un origen hasta un destino, al igual que Internet trasporta un paquete desde un origen hasta un destino. Pero ésta no es la analogía que estábamos buscando. Lo que queremos es encontrar una cierta *estructura* en la Figura 1.21. Si nos fijamos en esta figura, observaremos que hay una función Billete en cada extremo; también existe una función Equipaje

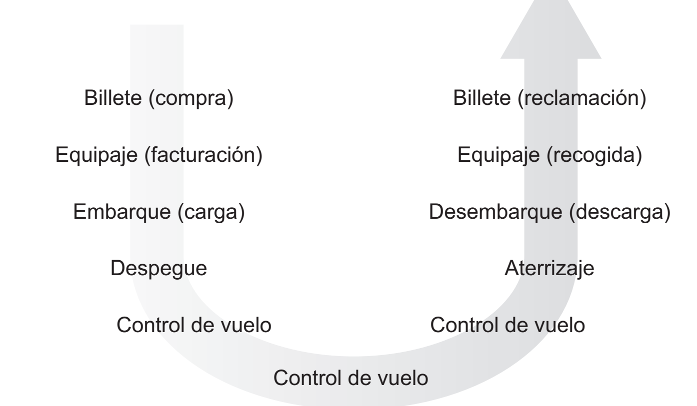


Figura 1.21 • Acciones para realizar un viaje en avión.

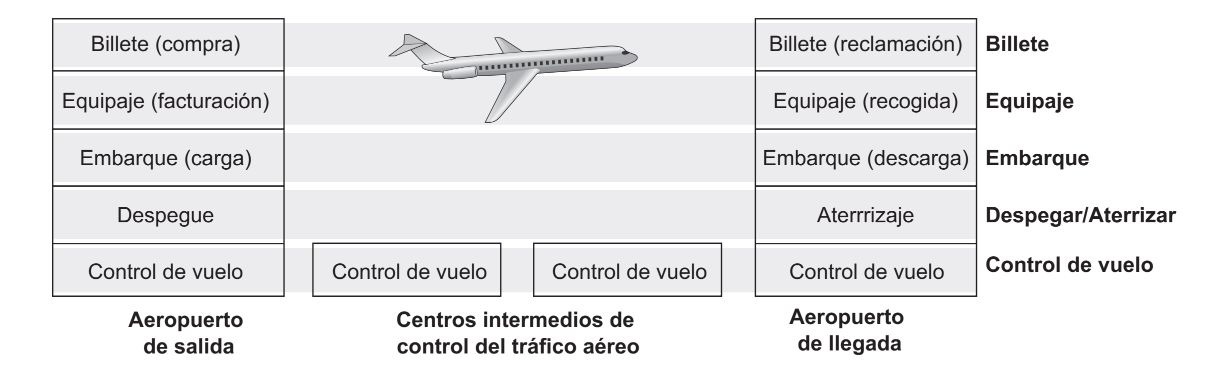


Figura 1.22 • Disposición de capas en horizontal de las funcionalidades de una compañía área.

para los pasajeros que tienen un billete y una función Embarque para los pasajeros que tienen billete y han facturado su equipaje. Para los pasajeros que han embarcado (es decir, que han adquirido un billete, han facturado las maletas y han embarcado), existe una función de despegue y aterrizaje y durante el vuelo, hay una función de control del avión. Esto sugiere que podemos fijarnos en las funcionalidades señaladas en la Figura 1.21 de forma *horizontal*, como se muestra en la Figura 1.22.

En la Figura 1.22 se han separado las distintas funciones de la compañía aérea en capas, proporcionando un marco de trabajo en el que podemos explicar cómo se realiza un viaje en avión. Observe que cada capa, combinada con las capas que tiene por debajo, implementa una cierta funcionalidad, un cierto *servicio.* En las capas Billete e inferiores se lleva a cabo la transferencia de una persona de un mostrador de línea aérea a otro. En las capas Equipaje e inferiores se realiza la transferencia de una persona y su equipaje dese el punto de facturación hasta la recogida de equipaje. Observe que la capa Equipaje sólo proporciona este servicio a las personas que ya han adquirido un billete. En la capa Embarque, se realiza la transferencia embarque/desembarque de una persona y su equipaje. En la capa Despegue/Aterrizaje, se realiza la transferencia pista a pista de personas y equipajes. Cada capa proporciona su servicio (1) llevando a cabo determinadas acciones dentro de dicha capa (por ejemplo, en la capa Embarque, se hace subir y bajar al pasaje del avión) y (2) utilizando los servicios de la capa que tiene directamente debajo de ella (por ejemplo, en la capa Embarque, utilizando el servicio de transferencia de pasajeros pista a pista de la capa Despegue/Aterrizaje).

Una arquitectura de capas nos permite estudiar una parte específica y bien definida de un sistema más grande y complejo. Esta simplificación por sí misma tiene un valor considerable al proporcionar modularidad, haciendo mucho más fácil modificar la implementación del servicio suministrado por la capa. Dado que la capa proporciona el mismo servicio a la capa que tiene por encima de ella y emplea los mismos servicios de la capa que tiene por debajo, el resto del sistema permanece invariable cuando se modifica la implementación de una capa. (Tenga en cuenta que cambiar la implementación de un servicio es muy diferente a cambiar el propio servicio.) Por ejemplo, si se modificara la función Embarque para que las personas embarcaran y desembarcaran por alturas, el resto del sistema de la compañía aérea no se vería afectado, ya que la capa Embarque continuaría llevando a cabo la misma función (cargar y descargar personas); simplemente implementa dicha función de una forma

diferente después de realizar el cambio. En sistemas complejos de gran tamaño que se actualizan constantemente, la capacidad de modificar la implementación de un servicio sin afectar al resto de los componentes del sistema es otra importante ventaja de la disposición en capas.

Capas de protocolos

Pero ya hemos hablado suficiente de compañías aéreas. Dirijamos ahora nuestra atención a los protocolos de red. Para proporcionar una estructura al diseño de protocolos de red, los diseñadores de redes organizan los protocolos (y el hardware y el software de red que implementan los protocolos) en **capas**. Cada protocolo pertenece a una de las capas, del mismo modo que cada función en la arquitectura de la compañía aérea de la Figura 1.22 pertenecía a una capa. De nuevo, estamos interesados en los **servicios** que ofrece una capa a la capa que tiene por encima, lo que se denomina **modelo de servicio** de capa. Como en el caso del ejemplo de la compañía aérea, cada capa proporciona su servicio (1) llevando a cabo ciertas acciones en dicha capa y (2) utilizando los servicios de la capa que tiene directamente debajo de ella. Por ejemplo, los servicios proporcionados por la capa n pueden incluir la entrega fiable de mensajes de un extremo de la red al otro. Esto podría implementarse mediante un servicio no fiable de entrega de mensajes terminal a terminal de la capa $n - 1$, y añadiendo la funcionalidad de la capa n para detectar y retransmitir los mensajes perdidos.

Una capa de protocolo puede implementarse por software, por hardware o mediante una combinación de ambos. Los protocolos de la capa de aplicación, como HTTP y SMTP, casi siempre se implementan por software en los sistemas terminales, al igual que los protocolos de la capa de transporte. Puesto que la capa física y las capas de enlace de datos son responsables de manejar la comunicación a través de un enlace específico, normalmente se implementan en las tarjetas de interfaz de red (por ejemplo, tarjetas Ethernet o WiFi) asociadas con un determinado enlace. La capa de red a menudo es una implementación mixta de hardware y software. Observe también que al igual que las funciones de la arquitectura de la compañía aérea estaban distribuidas entre los distintos aeropuertos y el centro de control de vuelo que formaban el sistema, también un protocolo de capa n está *distribuido* entre los sistemas terminales, los dispositivos de conmutación de paquetes y los restantes componentes que conforman la red. Es decir, suele haber una parte del protocolo de capa n en cada uno de estos componentes de red.

Las capas de protocolos presentan ventajas conceptuales y estructurales. Como hemos visto, las capas proporcionan una forma estructurada de estudiar los componentes del sistema. Además, la modularidad facilita la actualización de los componentes del sistema. Sin embargo, tenemos que comentar que algunos investigadores e ingenieros de redes se oponen vehemente a la estructura de capas [Wakeman 1992]. Un potencial inconveniente de la estructura de capas es que una capa puede duplicar la funcionalidad de la capa inferior. Por ejemplo, muchas pilas de protocolos proporcionan una función de recuperación de errores tanto por enlace como extremo a extremo. Un segundo potencial inconveniente es que la funcionalidad de una capa puede precisar información (por ejemplo, un valor de una marca temporal) que sólo existe en otra capa, y esto viola el objetivo de la separación en capas.

Cuando los protocolos de las distintas capas se toman en conjunto se habla de **pila de protocolos**. La pila de protocolos de Internet consta de cinco capas: capa física, capa de enlace, capa de red, capa de transporte y capa de aplicación, como se muestra en la

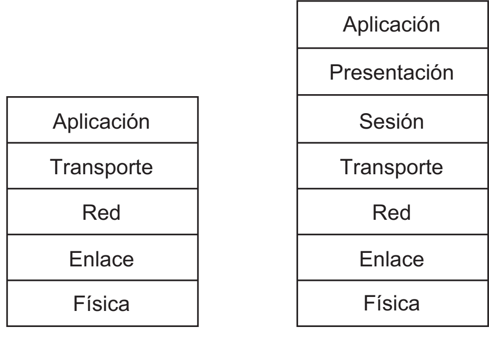


Figura 1.23 • Pila de protocolos de Internet (a) y modelo de referencia OSI (b).

Figura1.23(a). Si examina el Contenido, comprobará que hemos organizado el libro utilizando las capas de la pila de protocolos de Internet. Vamos a aplicar el **enfoque descendente**, abordando en primer lugar la capa de aplicación y continuando hacia abajo por la pila.

Capa de aplicación

La capa de aplicación es donde residen las aplicaciones de red y sus protocolos. La capa de aplicación de Internet incluye muchos protocolos, tales como el protocolo HTTP (que permite la solicitud y transferencia de documentos web), SMTP (que permite la transferencia de mensajes de correo electrónico) y FTP (que permite la transferencia de archivos entre dos sistemas terminales). Veremos que determinadas funciones de red, como la traducción de los nombres legibles que utilizamos las personas para los sistemas terminales de Internet (por ejemplo, www.ietf.org) en direcciones de red de 32 bits se realiza también con la ayuda de un protocolo específico de la capa de aplicación, en concreto, el Sistema de nombres de dominio (DNS, *Domain Name System*). En el Capítulo 2 veremos que es muy fácil crear e implantar nuestros propios protocolos de la capa de aplicación.

Un protocolo de la capa de aplicación está distribuido a lo largo de varios sistemas terminales, estando la aplicación en un sistema terminal que utiliza el protocolo para intercambiar paquetes de información con la aplicación de otro sistema terminal. A este paquete de información de la capa de aplicación lo denominaremos **mensaje**.

Capa de transporte

La capa de transporte de Internet transporta los mensajes de la capa de aplicación entre los puntos terminales de la aplicación. En Internet, existen dos protocolos de transporte, TCP y UDP, pudiendo cada uno de ellos transportar los mensajes de la capa de aplicación. TCP ofrece a sus aplicaciones un servicio orientado a la conexión. Este servicio proporciona un suministro garantizado de los mensajes de la capa de aplicación al destino y un mecanismo de control del flujo (es decir, adaptación de las velocidades del emisor y el receptor). TCP también divide los mensajes largos en segmentos más cortos y proporciona un mecanismo de control de congestión, de manera que un emisor regula su velocidad de

transmisión cuando la red está congestionada. El protocolo UDP proporciona a sus aplicaciones un servicio sin conexión. Es un servicio básico que no ofrece ninguna fiabilidad, ni control de flujo, ni control de congestión. En este libro, denominaremos a los paquetes de la capa de transporte **segmentos**.

Capa de red

La capa de red de Internet es responsable de trasladar los paquetes de la capa de red, conocidos como **datagramas**, de un host a otro. El protocolo de la capa de transporte (TCP o UDP) de Internet de un host de origen pasa un segmento de la capa de transporte y una dirección de destino a la capa de red, al igual que damos al servicio de correo postal una carta con una dirección de destino. Luego, la capa de red proporciona el servicio de suministrar el segmento a la capa de transporte del host de destino.

La capa de red de Internet incluye el conocido protocolo IP, que define los campos del datagrama, así como la forma en que actúan los sistemas terminales y los routers sobre estos campos. Existe un único protocolo IP y todos los componentes de Internet que tienen una capa de red deben ejecutar el protocolo IP. La capa de red de Internet también contiene los protocolos de enrutamiento que determinan las rutas que los datagramas siguen entre los orígenes y los destinos. Internet dispone de muchos protocolos de enrutamiento. Como hemos visto en la Sección 1.3, Internet es una red de redes y, dentro de una red, el administrador de la red puede ejecutar cualquier protocolo de enrutamiento que desee. Aunque la capa de red contiene tanto el protocolo IP como numerosos protocolos de enrutamiento, suele hacerse referencia a ella simplemente como la capa IP, lo que refleja el hecho de que IP es el pegamento que une todo Internet.

Capa de enlace

La capa de red de Internet encamina un datagrama a través de una serie de routers entre el origen y el destino. Para trasladar un paquete de un nodo (host o router) al siguiente nodo de la ruta, la capa de red confía en los servicios de la capa de enlace. En concreto, en cada nodo, la capa de red pasa el datagrama a la capa de enlace, que entrega el datagrama al siguiente nodo existente a lo largo de la ruta. En el siguiente nodo, la capa de enlace pasa el datagrama a la capa de red.

Los servicios proporcionados por la capa de enlace dependen del protocolo de la capa de enlace concreto que se emplee en el enlace. Por ejemplo, algunos protocolos de la capa de enlace proporcionan una entrega fiable desde el nodo transmisor, a través del enlace y hasta el nodo receptor. Observe que este servicio de entrega fiable es diferente del servicio de entrega fiable de TCP, que lleva a cabo una entrega fiable desde un sistema terminal a otro. Entre los ejemplos de protocolos de la capa de enlace se incluyen Ethernet, WiFi y el Protocolo punto a punto (PPP, *Point-to-Point Protocol*). Puesto que normalmente los datagramas necesitan atravesar varios enlaces para viajar desde el origen hasta el destino, un datagrama puede ser manipulado por distintos protocolos de la capa de enlace en los distintos enlaces disponibles a lo largo de la ruta. Por ejemplo, un datagrama puede ser manipulado por Ethernet en un enlace y por PPP en el siguiente enlace. La capa de red recibirá un servicio diferente por parte de cada uno de los distintos protocolos de la capa de enlace. En este libro, denominaremos a los paquetes de la capa de enlace **tramas**.

Capa física

Mientras que el trabajo de la capa de enlace es mover las tramas completas de un elemento de la red hasta el elemento de red adyacente, el trabajo de la capa física es el de mover los *bits individuales* dentro de la trama de un nodo al siguiente. Los protocolos de esta capa son de nuevo dependientes del enlace y, por tanto, dependen del medio de transmisión del enlace (por ejemplo, cable de cobre de par trenzado o fibra óptica monomodo). Por ejemplo, Ethernet dispone de muchos protocolos de la capa física: uno para cable de cobre de par trenzado, otro para cable coaxial, otro para fibra, etc. En cada caso, los bits se desplazan a través del enlace de forma diferente.

El modelo OSI

Una vez vista en detalle la pila de protocolos de Internet, deberíamos mencionar que no es la única pila de protocolos existente. En concreto, a finales de la década de 1970, la Organización Internacional de Estandarización (ISO, *International Organization for Standardization*) propuso que las redes de computadoras fueran organizadas utilizando siete capas, en lo que se vino a denominar modelo OSI (*Open Systems Interconnection*, Interconexión de sistemas abiertos) [ISO 2009]. El modelo OSI tomó forma cuando los protocolos que se convertirían en los protocolos de Internet estaban en su infancia y eran simplemente uno de los muchos conjuntos de protocolos diferentes que estaban en desarrollo; de hecho, probablemente los inventores del modelo OSI original no estaban pensando en Internet cuando lo crearon. No obstante, a partir de la década de 1970, muchos cursos universitarios y de formación, siguiendo las recomendaciones de ISO, organizaron cursos sobre el modelo de siete capas. A causa de su temprano impacto sobre la formación en redes, el modelo de siete capas todavía perdura en algunos libros de texto y cursos de formación sobre redes.

Las siete capas del modelo de referencia OSI, mostrado en la Figura 1.23(b), son: capa de aplicación, capa de presentación, capa de sesión, capa de transporte, capa de red, capa de enlace de datos y capa física. La funcionalidad de cinco de estas capas es básicamente la misma que sus contrapartidas del mismo nombre de Internet. Por tanto, vamos a centrarnos en las dos capas adicionales del modelo de referencia OSI: la capa de presentación y la capa de sesión. La función de la capa de presentación es la de proporcionar servicios que permitan a las aplicaciones que se comunican interpretar el significado de los datos intercambiados. Estos servicios incluyen la compresión y el cifrado de los datos (funciones cuyos nombres son autoexplicativos), así como la descripción de los datos (lo que, como veremos en el Capítulo 9, libera a la aplicación de tener que preocuparse por el formato interno en el que los datos se representan y almacenan, formatos que pueden diferir de una computadora a otra). La capa de sesión permite delimitar y sincronizar el intercambio de datos, incluyendo los medios para crear un punto de restauración y un esquema de recuperación.

El hecho de que en Internet falten dos de las capas existentes en el modelo de referencia OSI plantea un par de cuestiones interesantes: ¿acaso los servicios proporcionados por estas dos capas no son importantes? ¿Qué ocurre si una aplicación *necesita* uno de estos servicios? La respuesta de Internet a ambas preguntas es la misma: es problema del desarrollador de la aplicación. El desarrollador de la aplicación tiene que decidir si un servicio es importante y si lo es, será su problema el incorporar dicha funcionalidad a la aplicación.

1.5.2 Mensajes, segmentos, datagramas y tramas

La Figura 1.24 muestra la ruta física que siguen los datos al descender por la pila de protocolos de un sistema terminal emisor, al ascender y descender por las pilas de protocolos de un switch de la capa de enlace y de un router, para finalmente ascender por la pila de protocolos del sistema terminal receptor. Como veremos más adelante en el libro, los routers y los switches de la capa de enlace operan como dispositivos de conmutación de paquetes. De forma similar a los sistemas terminales, los routers y los switches de la capa de enlace organizan su hardware y software de red en capas. Pero estos dispositivos no implementan todas las capas de la pila de protocolos; habitualmente sólo implementan las capas inferiores. Como se muestra en la Figura 1.24, los switches de la capa de enlace implementan las capas 1 y 2; y los routers implementan las capas 1 a 3. Esto significa, por ejemplo, que los routers de Internet son capaces de implementar el protocolo IP (un protocolo de la capa 3) y los switches de la capa de enlace no. Veremos más adelante que aunque los switches de la capa de enlace no reconocen las direcciones IP, pueden reconocer las direcciones de la capa 2, como por ejemplo las direcciones Ethernet. Observe que los hosts implementan las cinco capas, lo que es coherente con la idea de que la arquitectura de Internet es mucho más compleja en las fronteras de la red.

La Figura 1.24 también ilustra el importante concepto de **encapsulación**. En el host emisor, un **mensaje de la capa de aplicación** (M en la Figura 1.24) se pasa a la capa de transporte. En el caso más simple, la capa de transporte recibe el mensaje y añade información

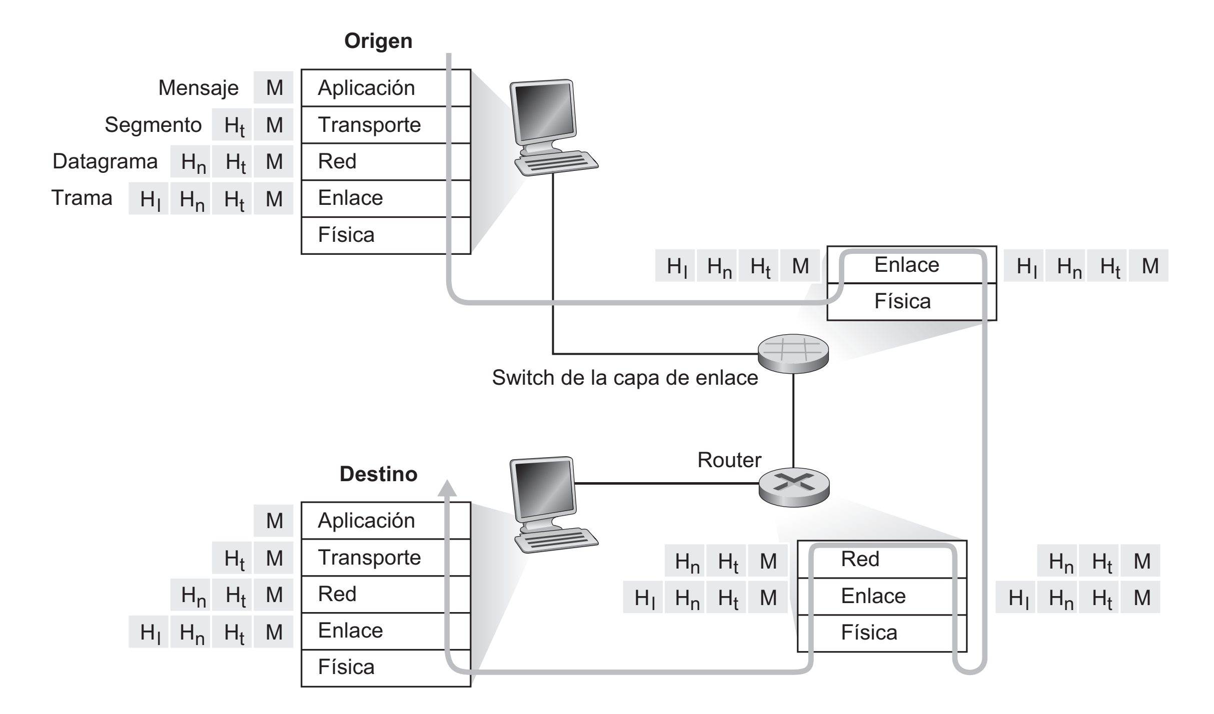


Figura 1.24 • Hosts, routers y switches de la capa de enlace. Cada uno de ellos contiene un conjunto distinto de capas, lo que refleja sus distintas funcionalidades.

adicional (denominada información de cabecera de la capa de transporte, H_t en la Figura 1.24) que será utilizada por la capa de transporte del lado receptor. El mensaje de la capa de aplicación y la información de cabecera de la capa de transporte constituyen el **segmento de la capa de transporte**. El segmento de la capa de transporte encapsula el mensaje de la capa de aplicación. La información añadida debe incluir información que permita a la capa de transporte del lado receptor entregar el mensaje a la aplicación apropiada y los bits de detección de errores que permitan al receptor determinar si los bits del mensaje han cambiado a lo largo de la ruta. A continuación, la capa de transporte pasa el segmento a la capa de red, que añade información de cabecera de la capa de red (H_n en la Figura 1.24) como son las direcciones de los sistemas terminales de origen y de destino, creando un **datagrama de la capa de red**. Este datagrama se pasa entonces a la capa de enlace, que (¡por supuesto!) añadirá su propia información de cabecera dando lugar a una **trama de la capa de enlace**. Así, vemos que en cada capa, un paquete está formado por dos tipos de campos: los campos de cabecera y un **campo de carga útil**. Normalmente, la carga útil es un paquete de la capa superior.

Una buena analogía para ilustrar este tema sería el envío de un informe interno de empresa desde una sucursal a otra a través del servicio postal público. Supongamos que Alicia, que se encuentra en una sucursal, quiere enviar un informe a Benito, que se encuentra en la otra sucursal. El *informe* es análogo al *mensaje de la capa de aplicación*. Alicia introduce el informe en un sobre para correo interno de la empresa y escribe en él el nombre y el número de departamento de Benito. El *sobre para correo interno* es análogo a un *segmento de la capa de transporte* (contiene la información de cabecera, el nombre y el departamento de Benito) y encapsula el mensaje de la capa de aplicación (el informe). Cuando en la sala de correo de la sucursal emisora se recibe este sobre, lo meten en otro sobre adecuado para enviar el informe mediante el servicio público de correos. En la sala de correo de la empresa también se escriben en el segundo sobre las direcciones postales tanto de la sucursal emisora como de la sucursal receptora. Aquí, el *sobre postal* es análogo al *datagrama*, encapsula el segmento de la capa de transporte (el sobre para correo interno), que encapsula el mensaje original (el informe). El servicio postal entrega el sobre postal a la sala de correos de la sucursal receptora, donde comienza el proceso de desencapsulación. En esta sala se extrae el informe y se envía a Benito. Por último, Benito abre el sobre para correo interno y saca el informe.

El proceso de encapsulación puede ser más complejo que el que acabamos de describir. Por ejemplo, un mensaje largo puede dividirse en varios segmentos de la capa de transporte (los cuales a su vez pueden dividirse en varios datagramas de la capa de red). En el extremo receptor, cada segmento tiene entonces que ser reconstruido a partir de sus datagramas constituyentes.

1.6 Ataques a las redes

Internet se ha convertido en una herramienta crítica para muchas instituciones actuales, incluyendo empresas pequeñas y medianas, universidades y organismos gubernamentales. Muchas personas individuales también confían en Internet para llevar a cabo muchas de sus actividades profesionales, sociales y personales. Pero detrás de todas estas utilidades y emociones, hay un lado oscuro, un lado donde los “chicos malos” intentan hacer estragos en nuestras vidas diarias dañando nuestras computadoras conectadas a Internet, violando nues-

tra privacidad y volviendo inoperables los servicios de Internet de los que dependemos [Skoudis 2006].

El campo de la seguridad de red se ocupa de ver cómo los "chicos malos" pueden atacar a las redes de computadoras y de cómo nosotros, que pronto seremos expertos en redes, podemos defendernos frente a estos ataques, o mejor todavía, de cómo diseñar nuevas arquitecturas que sean inmunes a tales ataques. Dada la frecuencia y variedad de ataques existentes, así como la amenaza de nuevos y más destructivos ataques futuros, la seguridad de red se ha convertido en un tema principal en el campo de las redes de comunicaciones en los últimos años. Una de las características de este libro de texto es que lleva las cuestiones sobre la seguridad en las redes a primer plano. En esta sección comenzaremos nuestra incursión en el campo de la seguridad de red, describiendo brevemente algunos de los ataques actuales más dañinos y predominantes en Internet. A continuación, en los siguientes capítulos nos ocuparemos en detalle de las tecnologías y los protocolos de red y consideraremos los diversos problemas relacionados con la seguridad, asociados con dichas tecnologías y protocolos. Por último, en el Capítulo 8, armados con nuestra experiencia recién adquirida en las redes de computadoras y los protocolos de Internet, estudiaremos en profundidad cómo las redes se pueden defender frente a los ataques, o diseñarse y operar para hacer que esos ataques sean imposibles en primera instancia.

Puesto que todavía no tenemos experiencia ni en redes ni en los protocolos de Internet, comenzaremos haciendo un repaso de algunos de los problemas de seguridad que predominan en la actualidad, con el fin de abrir boca para las explicaciones más sustanciosas que proporcionaremos en los capítulos siguientes. Así que podemos preguntarnos, ¿qué es lo que no funciona? ¿En qué sentido son vulnerables las redes de computadoras? ¿Cuáles son los principales tipos de ataques hoy día?

Los "malos" pueden introducir software malicioso en su host a través de Internet

Conectamos nuestros dispositivos a Internet porque deseamos recibir y enviar datos a Internet, lo que incluye todo tipo de cosas, páginas web, mensajes de correo electrónico, archivos MP3, llamadas telefónicas, vídeos en directo, resultados de motores de búsqueda, etc. Pero, lamentablemente, junto con todos estos elementos beneficiosos también existen elementos maliciosos, lo que se conoce de forma colectiva como **software malicioso** o **malware**, que puede acceder a nuestros dispositivos e infectarlos. Una vez que el malware ha infectado un dispositivo puede hacer todo tipo de maldades, como borrar nuestros archivos, instalar software espía que recopile nuestra información personal, como el número de la seguridad social, contraseñas y pulsaciones de teclas y luego enviar estos datos (a través de Internet, por supuesto) a los "chicos malos", a los atacantes. Nuestro host comprometido también puede pertenecer a una red de miles de dispositivos comprometidos de forma similar, lo que se conoce de forma colectiva como **botnet** (red robot), que los atacantes controlan y aprovechan para la distribución de correo electrónico basura (*spam*) o para llevar a cabo ataques distribuidos de denegación de servicio (que pronto explicaremos) contra los hosts objetivo.

Gran parte del malware que existe actualmente es **auto-replicante**: una vez que infecta un host, busca cómo acceder desde dicho host a otros hosts a través de Internet, y de nuevo desde esos hosts que acaba de infectar, busca cómo acceder a otros. De esta forma, el malware auto-replicante puede extenderse rápidamente de forma exponencial. Por ejemplo, el número de dispositivos infectados por el gusano 2003 Saphire/Slammer se replicaba cada

8,5 segundos en los primeros minutos después del brote, infectando más del 90 por ciento de los hosts vulnerables en 10 minutos [Moore 2003]. El malware puede extenderse en forma de virus, de un gusano o de un caballo de Troya [Skoudis 2004]. Un **virus** es un software malicioso que requiere cierta interacción del usuario para infectar el dispositivo. El ejemplo clásico es un adjunto de correo electrónico que contiene código ejecutable malicioso. Si un usuario recibe y abre un adjunto de este tipo, el usuario inadvertidamente ejecutará el malware en el dispositivo. Normalmente, tales virus enviados en los mensajes de correo electrónico se replican a sí mismos: una vez que se ha ejecutado, el virus puede enviar un mensaje idéntico con el mismo adjunto malicioso a, por ejemplo, todos los contactos de su libreta de direcciones. Un **gusano** (como el gusano Slammer) es malware que puede entrar en un dispositivo sin que el usuario interaccione de forma explícita. Por ejemplo, un usuario puede estar ejecutando una aplicación de red vulnerable a la que un atacante puede enviar software malicioso. En algunos casos, sin que el usuario intervenga, la aplicación puede aceptar el malware de Internet y ejecutarlo, creando un gusano. El gusano instalado ahora en el dispositivo recién infectado explora entonces Internet, buscando otros hosts que ejecuten la misma aplicación de red vulnerable. Cuando encuentra otros hosts vulnerables, envía una copia de sí mismo a esos hosts. Por último, un **caballo de Troya** es un malware que está oculto dentro de otro software que es útil. Hoy día, el malware está generalizado y es difícil defenderse de él. A lo largo del libro, le animaremos a que piense en la siguiente cuestión: ¿qué pueden hacer los diseñadores de redes de computadoras para defender a los dispositivos conectados a Internet de los ataques de malware?

Los “malos” pueden atacar a los servidores y a la infraestructura de red

Muchas de las amenazas de seguridad pueden clasificarse como **ataques de denegación de servicio (DoS, *Denial-of-Service*)**. Como su nombre sugiere, un ataque DoS vuelve inutilizable una red, un host o cualquier otro elemento de la infraestructura para los usuarios legítimos. Los servidores web, los servidores de correo electrónico, los servidores DNS (que se estudian en el Capítulo 2) y las redes institucionales pueden ser todos ellos objeto de ataques DoS. Los ataques DoS de Internet son muy comunes, teniendo lugar miles de ataques de este tipo cada año [Moore 2001; Mirkovic 2005]. La mayoría de los ataques DoS de Internet pueden clasificarse dentro de una de las tres categorías siguientes:

- *Ataque de vulnerabilidad.* Este ataque implica el envío de unos pocos mensajes bien construidos a una aplicación o sistema operativo vulnerable que esté ejecutándose en un host objetivo. Si se envía la secuencia de paquetes correcta a una aplicación o un sistema operativo vulnerable, el servicio puede detenerse o, lo que es peor, el host puede sufrir un fallo catastrófico.
- *Inundación del ancho de banda.* El atacante envía una gran cantidad de paquetes al host objetivo, de modo que comienzan a inundar el enlace de acceso del objetivo, impidiendo que los paquetes legítimos puedan alcanzar al servidor.
- *Inundación de conexiones.* El atacante establece un gran número de conexiones TCP completamente abiertas o semi-abiertas (estas conexiones se estudian en el Capítulo 3) en el host objetivo. El host puede comenzar a atascarse con estas conexiones fraudulentas impidiéndose así que acepte las conexiones legítimas.

Vamos a ver a continuación el ataque por inundación del ancho de banda más detalladamente. Recuerde el análisis que hemos realizado en la Sección 1.4.2 sobre los retardos y la

pérdida de paquetes; es evidente que si el servidor tiene una velocidad de acceso de *R* bps, entonces el atacante tendrá que enviar el tráfico a una velocidad de aproximadamente *R* bps para causar daños. Si *R* es muy grande, es posible que un único origen de ataque no sea capaz de generar el tráfico suficiente como para dañar al servidor. Además, si todo el tráfico procede de un mismo origen, un router situado en un punto anterior de la ruta puede detectar el ataque y bloquear todo el tráfico procedente de ese origen antes de que llegue cerca del servidor. En un ataque DoS distribuido (**DDoS, *Distributed DoS***), como el mostrado en la Figura 1.25, el atacante controla varios orígenes y hace que cada uno de ellos bombardee el objetivo con tráfico. Con este método, la tasa acumulada de tráfico para todos los orígenes controlados tiene que ser aproximadamente igual a *R* para inutilizar el servicio. Actualmente, se producen de forma común ataques DDoS que utilizan botnets con miles de hosts comprometidos [Mirkovic 2005]. Los ataques DDoS son mucho más difíciles de detectar y es mucho más complicado defenderse de ellos que de los ataques DoS procedentes de un único host.

Le animamos a que piense en la siguiente pregunta según vaya leyendo el libro: ¿qué pueden hacer los diseñadores de redes de computadoras para defenderlas de los ataques DoS? Veremos que son necesarias diferentes defensas para cada uno de los tres tipos de ataques DoS.

Los "malos" pueden examinar y analizar los paquetes

Actualmente, muchos usuarios acceden a Internet a través de dispositivos inalámbricos, tales como computadoras portátiles con conexión WiFi o dispositivos de mano con conexiones Internet móviles (lo que veremos en el Capítulo 6). Aunque el omnipresente acceso a Internet es extremadamente útil y habilita maravillosas aplicaciones para los usuarios móviles, también crea una importante vulnerabilidad de seguridad, al colocar un receptor pasivo en las vecindades del transmisor inalámbrico, que puede recibir una copia de todos los paquetes que se están transmitiendo. Estos paquetes pueden contener todo tipo de información confidencial, incluyendo contraseñas, números de la seguridad social, secretos comerciales

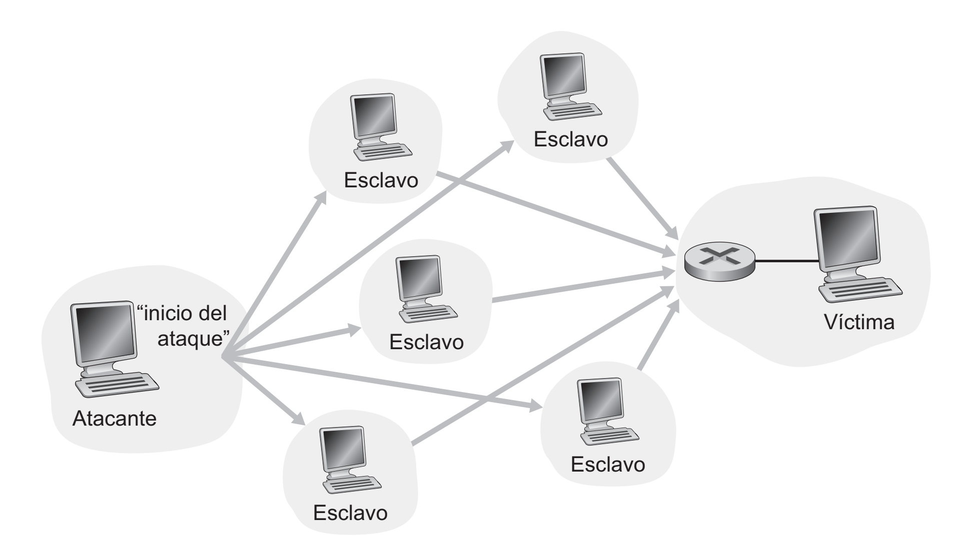


Figura 1.25 • Ataque de denegación de servicio distribuido.

y mensajes personales privados. Un receptor pasivo que registra una copia de todos los paquetes que pasan por él se conoce como **packet sniffer** (husmeador de paquetes).

Los sniffers también pueden implantarse en entornos cableados. En entornos cableados de multidifusión, como en muchas redes LAN Ethernet, un sniffer puede obtener copias de todos los paquetes enviados a través de la LAN. Como se ha descrito en la Sección 1.2, las tecnologías de acceso por cable también difunden paquetes y por tanto son vulnerables a ser monitorizados y analizados. Además, un atacante que consigue acceder al enlace de acceso o al router de acceso de una institución puede colocar un sniffer que haga una copia de todos los paquetes entrantes y salientes de la organización. Los paquetes así monitorizados pueden ser analizados después para obtener la información confidencial.

El software sniffer está disponible de forma gratuita en varios sitios web y como productos comerciales. Los profesores que imparten cursos sobre redes suelen realizar prácticas de laboratorio que implican escribir un programa sniffer y un programa de reconstrucción de datos de la capa de aplicación. Por supuesto, las prácticas de laboratorio con Wireshark [Wireshark 2009] asociadas con este texto (véase la práctica de laboratorio introductoria de Wireshark al final del capítulo) utilizan exactamente un programa sniffer así para monitorizar y analizar paquetes.

Puesto que los programas sniffer son pasivos, es decir, no inyectan paquetes en el canal, son difíciles de detectar. Por tanto, cuando envíamos paquetes a un canal inalámbrico, tenemos que aceptar que existe la posibilidad de que algún atacante pueda registrar copias de nuestros paquetes. Como es posible que haya adivinado, una de las mejores formas de defenderse frente a los programas sniffer son las técnicas criptográficas. En el Capítulo 8 veremos cómo se aplica la criptografía a la seguridad de la red.

Los “malos” pueden suplantar identidades

Es sorprendentemente fácil (y el lector tendrá los conocimientos necesarios para hacerlo muy pronto a medida que vaya leyendo este texto) crear un paquete con una dirección de origen, un contenido de paquete y una dirección de destino arbitrarios y luego transmitir dicho paquete a Internet, que reenviará el paquete a su destino. Imagine que el receptor confiado (por ejemplo, un router de Internet) que recibe tal paquete, toma la dirección de origen (falsa) como buena y luego ejecuta algún comando integrado en el contenido del paquete (por ejemplo, modifica la tabla de reenvío). La capacidad para inyectar paquetes en Internet con una dirección de origen falsa se conoce como **suplantación IP** y es una de las muchas formas en las que un usuario puede hacerse pasar por otro.

Para resolver este problema, necesitaremos aplicar un medio de *autenticación en el punto terminal,* es decir, un mecanismo que nos permita determinar con seguridad si un mensaje tiene su origen donde creemos que lo tiene. De nuevo, animamos a los lectores a que, a medida que avanzan por los capítulos del libro, piensen en cómo pueden hacer esto las aplicaciones y protocolos de red. En el Capítulo 8 exploraremos los mecanismos de autenticación en el punto terminal.

Los “malos” pueden modificar o borrar los mensajes

Terminamos este breve repaso sobre los ataques de red describiendo los **ataques de interposición**. En este tipo de ataques, los atacantes se introducen en la ruta de comunicaciones existente entre dos entidades que han establecido una conexión. Sean por ejemplo estas entidades Alicia y Benito, que pueden ser personas reales o entidades de red, como por ejemplo dos routers o dos servidores de correo. El atacante podría ser por ejemplo un router del que

el atacante haya tomado el control en la ruta de comunicación, o un módulo software residente en uno de los hosts terminales en una capa inferior de la pila de protocolos. En los ataques de interposición, el atacante no sólo tiene la capacidad de examinar y analizar los paquetes que pasan entre Benito y Alicia, sino que también puede inyectar, modificar o borrar paquetes. En la jerga utilizada al hablar de la seguridad de las redes, se dice que un ataque de interposición puede comprometer la *integridad* de los datos enviados entre Alicia y Benito. Como veremos en el Capítulo 8, los mecanismos que proporcionan confidencialidad (protección frente al husmeo de los paquetes) y autenticación en el punto terminal (lo que permite al receptor verificar con certeza al originador del mensaje) no necesariamente proporcionan integridad de los datos. Por tanto, necesitaremos otras técnicas para proporcionar esta funcionalidad.

Para terminar con este sección, vale la pena comentar cuál es la razón de que Internet se haya convertido en un lugar inseguro. Básicamente, la respuesta es que Internet fue diseñada originalmente para ser insegura, ya que se basaba en el modelo de un "grupo de usuarios que confiaban entre sí conectados a una red transparente" [Blumenthal 2001], un modelo en el que (por definición) no había necesidad de pensar en la seguridad. Muchos aspectos de la arquitectura de Internet original reflejan esta idea de confianza mutua. Por ejemplo, la posibilidad de que un usuario envíe un paquete a cualquier otro usuario es la opción predeterminada, en lugar de ser una capacidad solicitada/concedida, al igual que lo normal es creer que la identidad del usuario es la que declara, en lugar de autenticarle por defecto.

Pero, actualmente Internet no implica realmente "usuarios de confianza mutua". Sin embargo, los usuarios de hoy día necesitan comunicarse aunque no necesariamente confíen entre sí, pueden desear comunicarse de forma anónima, pueden comunicarse indirectamente a través de terceros (por ejemplo, cachés web, que estudiaremos en el Capítulo 2, o asistentes de movilidad, que veremos en el Capítulo 6), y deben desconfiar del hardware, el software e incluso del aire a través del que se comunican. A lo largo del libro vamos a encontrarnos con muchos retos relacionados con la seguridad, buscaremos formas de defendernos frente a los sniffer, la suplantación de identidades en el punto terminal, los ataques de interposición, los ataques DDoS, el software malicioso, etc. Tenemos que tener presente que la comunicación entre usuarios de mutua confianza es la excepción más que la regla. ¡Bienvenido al mundo de las redes modernas de comunicaciones!

1.7 Historia de Internet y de las redes de computadoras

En las Secciones 1.1 a 1.6 se ha hecho una presentación de las tecnologías utilizadas en las redes de comunicaciones e Internet. Ahora ya sabe lo suficiente como para impresionar a sus familiares y amigos. Sin embargo, si realmente desea causar una gran impresión en la siguiente fiesta a la que asista, debería salpicar su discurso con algunos detalles interesantes acerca de la fascinante historia de Internet [Segaller 1998].

1.7.1 El desarrollo de la conmutación de paquetes: 1961-1972

Tanto Internet como las redes de computadoras de hoy día tienen sus inicios a principios de la década de 1960, cuando la red telefónica era la red de comunicaciones dominante

en el mundo. Recordemos de la Sección 1.3 que la red telefónica utiliza mecanismos de conmutación de circuitos para transmitir la información entre un emisor y un receptor, una opción apropiada en la que la voz se transmite a una velocidad constante entre el emisor y el receptor. Debido a la creciente importancia (y los enormes costes) de las computadoras en los primeros años de la década de 1960 y a la aparición de las computadoras de tiempo compartido, quizá fue natural (¡al menos retrospectivamente!) considerar la cuestión de cómo enlazar las computadoras con el fin de que pudieran ser compartidas entre usuarios geográficamente distribuidos. El tráfico generado por esos usuarios probablemente era a *ráfagas,* compuesto por periodos de actividad como el envío de un comando a una computadora remota, seguido de periodos de inactividad mientras se espera a obtener una respuesta o mientras se contempla la respuesta recibida.

Tres grupos de investigación repartidos por el mundo, cada uno de ellos ignorante de la existencia de los otros [Leiner 1998], comenzaron a trabajar en la conmutación de paquetes como en una alternativa eficiente y robusta a la conmutación de circuitos. El primer trabajo publicado sobre las técnicas de conmutación de paquetes fue el de Leonard Kleinrock [Kleinrock 1961; Kleinrock 1964], un estudiante graduado en el MIT. Basándose en la teoría de colas, el trabajo de Kleinrock demostraba de forma elegante la efectividad de la técnica de conmutación de paquetes para las fuentes que generaban tráfico a ráfagas. En 1964, Paul Baran [Baran 1964] en el Rand Institute había comenzado a investigar el uso de la conmutación de paquetes para las comunicaciones de voz seguras en redes militares y, en el National Physical Laboratory (NPL) de Inglaterra, Donald Davies y Roger Scantlebury también estaban desarrollando sus ideas acerca de la conmutación de paquetes.

Los trabajos realizados en el MIT, en el instituto Rand y en los laboratorios NPL establecieron las bases de la red Internet actual. Pero Internet también tiene una larga tradición de "hagámoslo y demostremos que funciona" que también se remonta a la década de 1960. J. C. R. Licklider [DEC 1990] y Lawrence Roberts, ambos colegas de Kleinrock en el MIT, dirigieron el programa de Ciencias de la Computación en la Agencia de Proyectos de Investigación Avanzada (ARPA, *Advanced Research Projects Agency*) de Estados Unidos. Roberts publicó un plan global para la red ARPAnet [Roberts 1967], la primera red de computadoras de conmutación de paquetes y un ancestro directo de la red Internet pública actual. Los primeros conmutadores de paquetes se conocían como procesadores de mensajes de interfaz (**IMP, *Interface Message Processors***), y el contrato para construir estos conmutadores fue concedido a la empresa BBN. El Día del Trabajo de 1969, se instaló el primer IMP en UCLA bajo la supervisión de Kleinrock y poco después se instalaron tres IMP adicionales en el Instituto de Investigación de Stanford (SRI) en UC Santa Barbara y en la Universidad de Utah (Figura 1.26). Hacia finales de 1969 estaba disponible la red precursora de Internet, que estaba formada por cuatro nodos. Kleinrock recuerda la primera vez que utilizó la red para llevar a cabo un inicio de sesión remoto desde UCLA al SRI, consiguiendo que el sistema fallara [Kleinrock 2004].

Hacia 1972, la red ARPAnet había crecido aproximadamente hasta 15 nodos y la primera demostración pública fue realizada por Robert Kahn en 1972 en la *International Conference on Computer Communications*. Se completó el primer protocolo host a host entre sistemas terminales de ARPAnet, conocido como el protocolo de control de red (NCP, *Network Control Protocol*) [RFC 001]. Disponiendo de un protocolo terminal a terminal, ahora podían escribirse aplicaciones. Ray Tomlinson de BBN escribió el primer programa de correo electrónico en 1972.



Figura 1.26 • Uno de los primeros procesadores de mensajes de interfaz (IMP) y L. Kleinrock (Mark J. Terrill, AP/Wide World Photos).

1.7.2 Redes propietarias e interredes: 1972-1980

La red inicial ARPAnet era una única red cerrada. Para establecer una comunicación con un host de la red ARPAnet, había que estar realmente conectado a otro IMP de ARPAnet. A mediados de la década de 1970, comenzaron a surgir otras redes de conmutación de paquetes autónomas además de ARPAnet, entre las que se incluyen las siguientes:

- ALOHANet, una red de microondas que enlazaba universidades de las islas Hawai [Abramson 1970], así como redes de conmutación de paquetes vía satélite de DARPA [RFC 829] y redes de conmutación de paquetes vía radio [Kahn 1978].
- Telenet, una red de conmutación de paquetes comercial de BBN basada en la tecnología ARPAnet.
- Cyclades, una red de conmutación de paquetes francesa dirigida por Louis Pouzin [Think 2009].

- Redes de tiempo compartido tales como Tymnet y la red de Servicios de información GE, entre otras, a finales de la década de 1960 y principios de la década de 1970 [Schwartz 1977].
- La red SNA de IBM (1969-1974), que constituía un desarrollo paralelo al de ARPAnet [Schwartz 1977].

El número de redes fue creciendo. Retrospectivamente, podemos ver que había llegado el momento de desarrollar una arquitectura completa para la interconexión de redes. El trabajo pionero sobre interconexión de redes (realizado bajo el patrocinio de la Agencia DARPA, *Defense Advanced Research Projects Agency*), que en esencia es la creación de una *red de redes,* fue realizado por Vinton Cerf y Robert Kahn [Cerf 1974]; el término *internetting* (interredes o interconexión de redes) fue acuñado para describir este trabajo.

Estos principios arquitectónicos fueron integrados en TCP. Sin embargo, las primeras versiones de TCP eran bastante distintas al TCP de hoy día. Esas primeras versiones combinaban una entrega fiable en secuencia de los datos mediante retransmisiones del sistema terminal (esta función todavía la realiza TCP hoy día) con funciones de reenvío (que actualmente son realizadas por IP). Los primeros experimentos con TCP, combinados con el reconocimiento de la importancia de disponer de un servicio de transporte terminal a terminal no fiable y sin control de flujo para aplicaciones tales como voz empaquetada, llevaron a la separación de IP de TCP y al desarrollo del protocolo UDP. Los tres protocolos clave de Internet que se emplean actualmente (TCP, UDP e IP) fueron concebidos a finales de la década de 1970.

Además de las investigaciones relativas a Internet de la agencia DARPA, también se llevaron a cabo muchas otras importantes actividades de red. En Hawai, Norman Abramson desarrolló ALOHAnet, una red de paquetes vía radio que permitía a varios sitios remotos de las islas Hawai comunicarse entre sí. El protocolo ALOHA [Abramson 1970] fue el primer protocolo de acceso múltiple, que permitió a usuarios distribuidos geográficamente compartir un mismo medio de comunicación de difusión (una frecuencia de radio). Metcalfe y Boggs se basaron en el protocolo de acceso múltiple de Abramson para desarrollar el protocolo Ethernet [Metcalfe 1976] para redes de difusión compartidas basadas en cable; véase la Figura 1.27. Es interesante comentar que el protocolo Ethernet de Metcalfe y Boggs fue motivado por la necesidad de conectar varios PC, impresoras y discos compartidos [Perkins 1994]. Hace veinticinco años, bastante antes de la revolución de los PC y de la explosión de las redes, Metcalfe y Boggs establecieron las bases para las redes LAN de computadoras PC actuales. La tecnología Ethernet también representó un paso importante en la interconexión de redes (interred). Cada red de área local Ethernet era una red por sí misma, y dado que el número de redes LAN proliferaba, la necesidad de interconectar estas redes LAN adquiría cada vez más importancia. En el Capítulo 5 veremos en detalle Ethernet, ALOHA y otras tecnologías LAN.

1.7.3 Proliferación de las redes: 1980-1990

A finales de la década de 1970, había unos doscientos hosts conectados a la red ARPAnet. A finales de la década de 1980, el número de hosts conectados a la red Internet pública, una confederación de redes similar a la Internet actual, llegaría a los cien mil. La década de 1980 fue una época de enorme crecimiento.

Gran parte de este crecimiento fue el resultado de varios y distintos esfuerzos por crear redes de computadoras que enlazaran universidades. BITNET proporcionaba servicios de

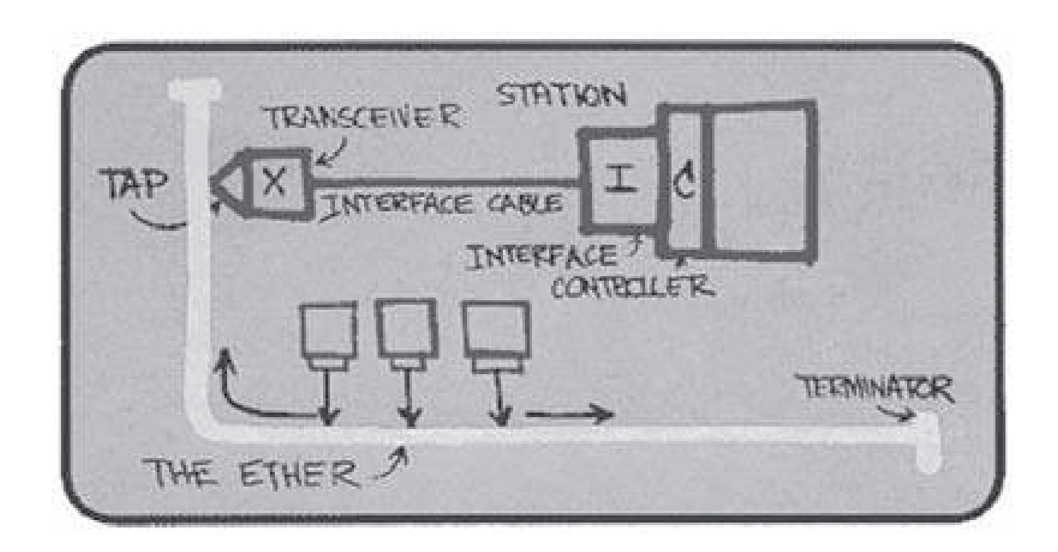


Figura 1.27 • Concepción original de Ethernet de Metcalfe.

correo electrónico y de transferencia de archivos entre varias universidades. CSNET (*Computer Science Network*) se formó para que investigadores universitarios que no tenían acceso a la red ARPAnet pudieran comunicarse. En 1986 se creó NSFNET para proporcionar acceso a los centros de supercomputación patrocinados por NSF. Inicialmente, con una velocidad en la red troncal de 56 kbps, la red troncal de NSFNET llegaría a operar a 1,5 Mbps a finales de la década y serviría como una red troncal primaria para enlazar redes regionales.

En la comunidad ARPAnet, muchas de las piezas finales de la arquitectura de Internet actual fueron encajando. El 1 de enero de 1983 se llevó a cabo el lanzamiento oficial de TCP/IP como el nuevo protocolo de host estándar para ARPAnet (reemplazando al protocolo NCP). La transición [RFC 801] de NCP a TCP/IP fue un suceso señalado: a todos los hosts se les requirió pasar a utilizar TCP/IP ese día. A finales de la década de 1980, se realizaron importantes extensiones en TCP con el fin de implementar el control de congestión basado en host [Jacobson 1988]. También se desarrolló el sistema DNS, que se emplea para establecer la correspondencia entre los nombres de Internet que usamos las personas (por ejemplo, gaia.cs.umass.edu) y sus direcciones IP de 32 bits [RFC 1034].

En paralelo con el desarrollo de ARPAnet (realizado fundamentalmente en Estados Unidos), a principios de la década de 1980, los franceses lanzaron el proyecto Minitel, un ambicioso plan para llevar las redes de datos a todos los hogares. Patrocinado por el gobierno francés, el sistema Minitel consistía en una red pública de conmutación de paquetes (basada en la serie de protocolos X.25), servidores Minitel y terminales económicos que incorporaban modems de baja velocidad. Minitel alcanzó un gran éxito en 1984 cuando el gobierno francés proporcionó un terminal Minitel gratuito a todo aquel que deseara tener uno en su casa. Los sitios Minitel incluían sitios gratuitos, como por ejemplo el directorio telefónico, y sitios privados, que cobraban sus tarifas basándose en la utilización del servicio. A mediados de la década de 1990, se produjo el pico de servicios ofertados ofreciendo más de 20.000 servicios, desde servicios de banca hasta bases de datos de investigación especializadas. Esta red era utilizada por más del 20 por ciento de la población de Francia,

generó más de 1.000 millones de dólares de ingresos anuales y dió lugar a la creación de 10.000 puestos de trabajo. Minitel se utilizó en muchos hogares de Francia 10 años antes de que los americanos ni siquiera hubieran oído hablar de Internet.

1.7.4 La explosión de Internet: década de 1990

La década de 1990 estuvo marcada por una serie de acontecimientos que simbolizaron la continua evolución y la pronta llegada de la comercialización de Internet. ARPAnet, el progenitor de Internet, dejó de existir. MILNET y la Red de Datos de Defensa habían crecido durante la década de 1980 transportando la mayor parte del tráfico relacionado con el Departamento de Defensa de Estados Unidos y NSFNET había empezado a servir como red troncal para conectar las redes regionales de Estados Unidos y las redes nacionales de ultramar. En 1991, NSFNET retiró sus restricciones sobre el uso de NSFNET para propósitos comerciales. La propia NSFNET fue eliminada del servicio activo en 1995, siendo el tráfico troncal de Internet transportado por los Proveedores de servicios de Internet comerciales.

Sin embargo, el principal acontecimiento de la década de 1990 fue la aparición de la World Wide Web, que llevaría Internet a los hogares y negocios de millones de personas de todo el mundo. La Web sirvió como plataforma para posibilitar e implantar cientos de nuevas aplicaciones, que hoy damos por sentadas. Si desea conocer una breve historia sobre la primera época de la Web, consulte [W3C 1995].

La Web fue inventada en el CERN por Tim Berners-Lee entre 1989 y 1991 [Berners-Lee 1989], basándose en las ideas de los primeros trabajos acerca del hipertexto desarrolladas en la década de 1940 por Vannevar Bush [Bush 1945] y luego en la década de 1960 por Ted Nelson [Xanadu 2009]. Berners-Lee y sus socios desarrollaron las versiones iniciales de HTML, HTTP, un servidor web y un navegador (los cuatro componentes clave de la Web). Hacia finales del año 1993 estaban operativos aproximadamente doscientos servidores web. Esta colección de servidores sólo sería un presagio de lo que estaba por venir. Al mismo tiempo, había varios investigadores desarrollando navegadores web con interfaces GUI, entre los que se encontraba Marc Andreessen, que lideró el desarrolló del popular navegador con interfaz GUI Mosaic. En 1994, Marc Andreessen y Jim Clark crearon Mosaic Communications, que más tarde se convertiría en Netscape Communications Corporation [Cusumano 1998; Quittner 1998]. Hacia 1995, los estudiantes universitarios empleaban los navegadores Mosaic y Netscape para navegar por la Web diariamente. También por esa época, las empresas, grandes y pequeñas, comenzaron a trabajar con servidores web y a realizar transacciones comerciales a través de la Web. En 1996, Microsoft empezó a desarrollar navegadores y comenzaría la guerra de los navegadores entre Netscape y Microsoft, que terminaría ganando Microsoft unos pocos años después [Cusumano 1998].

La segunda mitad de la década de 1990 fue un periodo de gran crecimiento e innovación para Internet, con las principales corporaciones y miles de empresas creando productos y servicios Internet. El correo electrónico a través de Internet continuó evolucionando con lectores de correo cada vez con más funcionalidades que proporcionaban libretas de direcciones, adjuntos, hipervínculos activos y transporte multimedia. Al final del milenio, Internet daba soporte a cientos de aplicaciones populares, entre las que se incluyen:

- Correo electrónico, incluyendo los adjuntos y el correo electrónico accesible a través de la Web.
- La Web, incluyendo la navegación web y el comercio por Internet.

- La mensajería instantánea, con listas de contactos, primeramente introducida por ICQ.
- La compartición igualitaria de archivos MP3, de la que Napster fue la pionera.

Es interesante saber que las dos primeras de esta aplicaciones estrella fueron creadas por la comunidad de investigadores, mientras que las dos últimas lo fueron por unos pocos jóvenes emprendedores.

El periodo comprendido entre 1995 y 2001 fue un paseo en montaña rusa para Internet en los mercados financieros. Antes de que fueran incluso rentables, cientos de nuevas empresas de Internet fueron vendidas en el mercado bursátil con oferta pública inicial. Muchas empresas fueron valoradas en miles de millones de dólares sin tener ingresos significativos. Las acciones de Internet se hundieron en 2000-2001 y muchas de esas empresas de nueva creación cerraron. No obstante, algunas de ellas emergieron como los grandes ganadores en el espacio Internet, entre las que se incluyen Microsoft, Cisco, Yahoo, e-Bay, Google y Amazon.

1.7.5 Desarrollos recientes

La innovación en las redes de computadoras continúa a buen paso. Se han hecho avances en todos los frentes, incluyendo la implementación de nuevas aplicaciones, distribución de contenidos, telefonía por Internet, velocidades de transmisión más altas en las redes LAN y routers más rápidos. Pero los tres desarrollos que merecen una atención especial son la proliferación de métodos de acceso de alta velocidad (incluido el acceso inalámbrico), la seguridad y las redes P2P.

Como se ha visto en la Sección 1.2, la creciente penetración del acceso a Internet residencial mediante módem por cable y DSL abre una etapa de abundancia para nuevas aplicaciones multimedia, incluyendo voz y vídeo sobre IP [Skype 2009], compartición de vídeos [YouTube 2009] y televisión sobre IP [PPLive 2009]. La creciente omnipresencia de las redes públicas Wi-Fi de alta velocidad (11 Mbps y superior) y el acceso a Internet a velocidades medias (cientos de kbps) mediante redes de telefonía móvil no hacen sólo posible estar constantemente conectado, sino que también habilitan un nuevo y excitante conjunto de servicios específicos de la ubicación. En el Capítulo 6 nos ocuparemos de las redes inalámbricas y de la movilidad.

Después de una serie de ataques de denegación de servicio en los servidores web más importantes a finales de la década de 1990 y de la proliferación de ataques mediante gusanos (como el gusano Blaster), la seguridad de las redes se convirtió en un tema extremadamente importante. Estos ataques dieron lugar al desarrollo de los sistemas de detección de intrusiones, que son capaces de advertir en una fase temprana de un ataque, y al uso de cortafuegos para filtrar el tráfico no deseado antes de que entre en la red. En el Capítulo 8 veremos unos cuantos temas importantes relacionados con la seguridad.

La última innovación de la que tomaremos nota son las redes P2P. Una aplicación de red P2P explota los recursos disponibles en las computadoras de los usuarios (almacenamiento, contenido, ciclos de CPU y presencia humana) y tiene una autonomía significativa con respecto a los servidores centrales. Normalmente, las computadoras de los usuarios (es decir, los pares) disponen de una conectividad intermitente. Son numerosas las historias de éxito P2P que han tenido lugar en los últimos años, entre las que se incluyen la compartición de archivos P2P (Napster, Kazaa, Gnutella, eDonkey, LimeWire, etc.), la distribución de

archivos (BitTorrent), Voz sobre IP (Skype) e IPTV (PPLive, ppStream). En el Capítulo 2 veremos muchas de estas aplicaciones P2P.

1.8 Resumen

En este capítulo hemos presentado una gran cantidad de material. Hemos hablado de las distintas piezas hardware y software que forman Internet en particular y las redes de computadoras en general. Hemos partido de la frontera de la red, fijándonos en los sistemas terminales y las aplicaciones, y en el servicio de transporte proporcionado a las aplicaciones que se ejecutan en los sistemas terminales. También hemos hablado de las tecnologías de la capa de enlace y de los medios físicos que pueden encontrarse normalmente en la red de acceso. A continuación, nos hemos adentrado en el interior de la red, en su núcleo, y hemos identificado los mecanismos de conmutación de paquetes y de conmutación de circuitos como los dos métodos básicos utilizados para el transporte de datos a través de una red de telecomunicaciones, y hemos examinado las fortalezas y las debilidades de cada método. También hemos examinado la estructura global de Internet, y hemos aprendido que es una red de redes. Hemos visto que la estructura jerárquica de Internet, formada por los ISP de nivel superior e inferior, ha permitido que Internet crezca hasta incluir miles de redes.

En la segunda parte de este capítulo de introducción, hemos abordado varios temas fundamentales del campo de las redes. En primer lugar, hemos estudiado las causas de los retardos, la tasa de transferencia y la pérdida de paquetes en una red de conmutación de paquetes. Hemos desarrollado modelos cuantitativos simples para determinar los retardos de transmisión, de propagación y de cola, así como para la tasa de transferencia; hemos hecho un uso exhaustivo de estos modelos de retardo en los problemas de repaso incluidos al final del capítulo. A continuación, hemos visto las capas de protocolos y los modelos de servicio, las principales claves arquitectónicas de las redes a las que se volverá a hacer referencia a lo largo del libro. Asimismo, hemos repasado los ataques de seguridad más habituales actualmente en Internet. Hemos terminado nuestra introducción con un breve repaso a la historia de las redes de computadoras. Este primer capítulo en sí mismo constituye un minicurso sobre redes de computadoras.

Por tanto, hemos cubierto una enorme cantidad de conceptos básicos en este primer capítulo. Si está un poco abrumado, no se preocupe, en los siguientes capítulos revisaremos todas estas ideas, viéndolas con mucho más detalle (lo que es una promesa, no una amenaza). En este punto, esperamos que termine el capítulo teniendo ya una idea sobre las piezas que forman una red, teniendo un conocimiento (que todavía deberá desarrollar) del vocabulario del campo de las redes de computadoras (no se asuste por tener que volver a consultar este capítulo) y habiendo incrementado su deseo por aprender más acerca de las redes. Ésta es la tarea que tenemos por delante durante el resto del libro.

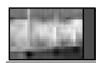
Mapa de este libro

Antes de iniciar cualquier viaje, siempre debería echarse un vistazo a un mapa de carreteras para familiarizarse con las principales carreteras y desvíos con los que nos encontraremos más adelante. En el viaje en el que nos hemos embarcado nosotros, el destino final es conocer en profundidad el cómo, el qué y el por qué de las redes de computadoras, y nuestro mapa son los capítulos de este libro:

1. Redes de computadoras e Internet
2. La capa de aplicación
3. La capa de transporte
4. La capa de red
5. La capa de enlace y las redes de área local
6. Redes inalámbricas y móviles
7. Redes multimedia
8. Seguridad en las redes de computadoras
9. Gestión de redes

Los Capítulos 2 a 5 son los cuatro capítulos centrales del libro. Observará que están organizados según las cuatro capas superiores de la pila de protocolos de Internet de cinco capas, un capítulo por capa. Observará también que nuestro viaje va a comenzar por la parte superior de la pila de protocolos de Internet, es decir, por la capa de aplicación, y luego continuaremos nuestro trabajo descendiendo por la pila. El razonamiento de hacer este recorrido de arriba-abajo es porque una vez que conozcamos las aplicaciones, estaremos en condiciones de comprender los servicios de red necesarios para dar soporte a esas aplicaciones. Podremos así examinar las distintas formas en que tales servicios pueden ser implementados por una arquitectura de red. Ocuparnos de las aplicaciones en primer lugar nos va a proporcionar la motivación necesaria para abordar el resto del texto.

En la segunda mitad del libro, los Capítulos 6 a 9, se abordan cuatro temas enormemente importantes (y en cierto modo independientes) en las redes de comunicaciones modernas. En el Capítulo 6, examinaremos las redes inalámbricas y celulares, incluyendo las redes LAN inalámbricas (lo que incluye las tecnologías WiFi, WiMAX y Bluetooth), las redes de telefonía móvil (lo que incluye las redes GSM) y la movilidad (tanto en redes IP como GSM). En el Capítulo 7, “Redes multimedia”, examinaremos aplicaciones de audio y de vídeo tales como la telefonía Internet, la videoconferencia y los flujos de información multimedia almacenada. También veremos cómo pueden diseñarse las redes de conmutación de paquetes para proporcionar una calidad de servicio coherente a las aplicaciones de audio y de vídeo. En el Capítulo 8, “Seguridad en las redes de computadoras”, veremos en primer lugar los fundamentos de los mecanismos de cifrado y de seguridad de red y después examinaremos cómo está siendo aplicada la teoría básica a un amplio rango de contextos de Internet. El último capítulo, “Gestión de redes”, examina los problemas fundamentales de la administración de red, así como los principales protocolos de Internet empleados para administrar la red.



Problemas y cuestiones de repaso

Capítulo 1 Cuestiones de repaso

SECCIÓN 1.1

R1. ¿Cuál es la diferencia entre un host y un sistema terminal? Enumere los tipos de sistemas terminales. ¿Es un servidor web un sistema terminal?

R2. El término *protocol*o a menudo se emplea para describir las relaciones diplomáticas. Proporcione un ejemplo de un protocolo diplomático.

SECCIÓN 1.2

R3. ¿Qué es un programa cliente? ¿Qué es un programa servidor? ¿Un programa servidor solicita y recibe servicios de un programa cliente?

R4. Enumere seis tecnologías de acceso. Clasifíquelas como de acceso residencial, acceso empresarial o acceso móvil.

R5. ¿La velocidad de transmisión en un sistema HFC es dedicada o compartida entre los usuarios? ¿Pueden producirse colisiones en un canal de descarga HFC? ¿Por qué?

R6. Enumere las tecnologías de acceso residencial disponibles en su ciudad. Para cada tipo de acceso, detalle la velocidad de descarga, la velocidad de carga y el precio mensual.

R7. ¿Cuál es la velocidad de transmisión en las redes LAN Ethernet? Para una determinada velocidad de transmisión, ¿pueden los usuarios de la LAN transmitir continuamente a dicha velocidad?

R8. Cite algunos de los medios físicos sobre los que se puede emplear la tecnología Ethernet.

R9. Para el acceso residencial se emplean los modems de acceso telefónico, los sistemas HFC, DSL y FTTH. Para cada una de estas tecnologías de acceso, detalle el rango de velocidades de transmisión e indique si la velocidad de transmisión es dedicada o compartida.

R10. Describa las tecnologías de acceso inalámbrico a Internet más populares hoy día.

SECCIÓN 1.3

R11. ¿Qué ventajas presenta una red de conmutación de circuitos frente a una red de conmutación de paquetes? ¿Qué desventajas tiene la multiplexación TDM frente a la multiplexación FDM en una red de conmutación de circuitos?

R12. ¿Por qué se dice que la conmutación de paquetes emplea multiplexación estadística? Compare la multiplexación estadística con la multiplexación por división en el tiempo (TDM).

R13. Suponga que hay un único dispositivo de conmutación de paquetes entre un host emisor y un host receptor. Las velocidades de transmisión entre el host emisor y el dispositivo de conmutación (switch) y entre el switch y el host receptor son R_1 y R_2, respectivamente. Suponiendo que el switch utiliza el mecanismo de conmutación de paquetes de almacenamiento y reenvío, ¿cuál es el retardo total terminal a terminal si se envía un paquete de longitud L? (Ignore los retardos de cola, de propagación y de procesamiento.)

R14. ¿Cuál es la diferencia entre un ISP de nivel 1 y un ISP de nivel 2?

R15. Suponga que los usuarios comparten un enlace de 2 Mbps y que cada usuario transmite a una velocidad de 1 Mbps continuamente, pero sólo durante el 20 por ciento del tiempo. (Véase la explicación sobre la multiplexación estadística de la Sección 1.3.)

a. Si se utiliza la conmutación de circuitos, ¿a cuántos usuarios puede darse soporte?

b. Para el resto del problema, suponga que se utiliza la conmutación de paquetes. ¿Por qué prácticamente no habrá retardo de cola antes del enlace si dos o menos usuarios transmiten a un mismo tiempo? ¿Por qué existirá retardo de cola si tres usuarios transmiten simultáneamente?

c. Calcule la probabilidad de que un usuario dado esté transmitiendo.

d. Suponga ahora que hay tres usuarios. Calcule la probabilidad de que en un instante determinado los tres usuarios estén transmitiendo simultáneamente. Halle la fracción de tiempo durante la que la cola crece.

SECCIÓN 1.4

R16. Considere el envío de un paquete desde un host emisor a un host receptor a través de una ruta fija. Enumere los componentes del retardo terminal a terminal. ¿Cuáles de estos retardos son constantes y cuáles son variables?

R17. Visite el applet *Transmission Versus Propagation Delay* (transmisión frente a retardo de propagación) disponible en el sitio web del libro. Utilizando las velocidades, retardos de propagación y tamaños de paquete disponibles, determine una combinación para la cual el emisor termine la operación de transmisión antes de que el primer bit del paquete haya llegado al receptor. Halle otra combinación para la que el primer bit del paquete haga llegado al receptor antes de que el emisor haya terminado de transmitir.

R18. ¿Cuánto tiempo tarda un paquete cuya longitud es de 1.000 bytes en propagarse a través de un enlace a una distancia de 2.500 km, siendo la velocidad de propagación igual a $2{,}5 \cdot 10^8$ m/s y la velocidad de transmisión a 2 Mbps? De forma más general, ¿cuánto tiempo tarda un paquete de longitud L en propagarse a través de un enlace a una distancia d, con una velocidad de propagación s y una velocidad de transmisión de R bps? ¿Depende este retardo de la longitud del paquete? ¿Depende este retardo de la velocidad de transmisión?

R19. Suponga que el host A desea enviar un archivo de gran tamaño al host B. La ruta desde el host A al host B está formada por tres enlaces, cuyas velocidades son R_1 = 500 kbps, R_2 = 2 Mbps y R_3 = 1 Mbps.

a. Suponiendo que no hay tráfico en la red, ¿cuál es la tasa de transferencia para el archivo?

b. Suponga que el tamaño del archivo es de 4 millones de bytes. Dividiendo el tamaño del archivo entre la tasa de transferencia, ¿cuánto tiempo tardará aproximadamente en transferirse el archivo al host B?

c. Repita los apartados (a) y (b), pero ahora con R_2 igual a 100 kbps.

R20. Suponga que el sistema terminal A desea enviar un archivo de gran tamaño al sistema terminal B. Sin entrar en detalles, describa cómo crea el sistema terminal A los paquetes a partir del archivo. Cuando uno de estos paquetes llega a un conmutador de paquetes, ¿qué información del mismo utiliza el conmutador para determinar el enlace por el que debe ser reenviado el paquete? ¿Por qué la conmutación de paquetes en Internet es análoga a viajar de una ciudad a otra preguntando por la dirección a la que nos dirigimos?

R21. Visite el applet *Queuing and Loss* (colas y pérdida de paquetes) en el sitio web del libro. ¿Cuáles son las velocidades de transmisión máxima y mínima? Para esas velocidades, ¿cuál es la intensidad de tráfico? Ejecute el applet con esas velocidades y determine cuánto tiempo tiene que transcurrir para que tenga lugar una pérdida de paquetes. A continuación, repita el experimento una segunda vez y determine de nuevo cuánto

tiempo pasa hasta producirse una pérdida de paquetes. ¿Son diferentes los valores obtenidos? ¿Por qué?

SECCIÓN 1.5

R22. Enumere cinco tareas que puede realizar una capa. ¿Es posible que una (o más) de estas tareas pudieran ser realizadas por dos (o más) capas?

R23. ¿Cuáles son las cinco capas de la pila de protocolos Internet? ¿Cuáles son las responsabilidades principales de cada una de estas capas?

R24. ¿Qué es un mensaje de la capa de aplicación? ¿Y un segmento de la capa de transporte? ¿Y un datagrama de la capa de red? ¿Y una trama de la capa de enlace?

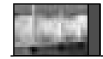
R25. ¿Qué capas de la pila de protocolos de Internet procesa un router? ¿Qué capas procesa un switch de la capa de enlace? ¿Qué capas procesa un host?

SECCIÓN 1.6

R26. ¿Cuál es la diferencia entre un virus, un gusano y un caballo de Troya?

R27. Describa cómo puede crearse una red robot (botnet) y cómo se puede utilizar en un ataque DDoS.

R28. Suponga que Alicia y Benito están enviándose paquetes entre sí a través de una red. Imagine que Victoria se introduce en la red de modo que puede capturar todos los paquetes enviados por Alicia y que luego envía lo que ella quiere a Benito. Además, también puede capturar todos los paquetes enviados por Benito y luego enviar a Alicia lo que le parezca. Enumere algunos de los daños que Victoria puede ocasionar desde su posición.



Problemas

P1. Diseñe y describa un protocolo de nivel de aplicación que será utilizado entre un cajero automático y la computadora central de un banco. El protocolo deberá permitir verificar la tarjeta y la contraseña del usuario, consultar el saldo de la cuenta (que se almacena en la computadora central) y hacer un apunte en la cuenta por la cantidad retirada por el usuario. Las entidades del protocolo deben poder controlar todos los casos en los que no hay suficiente saldo en la cuenta como para cubrir el reembolso. Especifique el protocolo enumerando los mensajes intercambiados y las acciones realizadas por el cajero automático o la computadora central del banco al transmitir y recibir mensajes. Haga un boceto del funcionamiento del protocolo para el caso de una retirada de efectivo sin errores, utilizando un diagrama similar al mostrado en la Figura 1.2. Establezca explícitamente las suposiciones hechas por el protocolo acerca del servicio de transporte terminal a terminal subyacente.

P2. Considere una aplicación que transmite datos a una velocidad constante (por ejemplo, el emisor genera una unidad de datos de N bits cada k unidades de tiempo, donde k es un valor pequeño y fijo). Además, cuando esta aplicación se inicia, se ejecutará durante un periodo de tiempo relativamente largo. Responda a las siguientes cuestiones de forma breve y justificando su respuesta:

a. ¿Qué sería más apropiado para esta aplicación, una red de conmutación de circuitos o una red de conmutación de paquetes? ¿Por qué?

b. Suponga que se utiliza una red de conmutación de paquetes y el único tráfico que existe en la misma procede de la aplicación que acabamos de describir. Además, suponga que la suma de las velocidades de datos de la aplicación es menor que las capacidades de cada uno de los enlaces. ¿Será necesario algún mecanismo de control de congestión? ¿Por qué?

P3. Considere la red de conmutación de circuitos de la Figura 1.12. Recuerde que hay n circuitos en cada enlace.

a. ¿Cuál es el número máximo de conexiones simultáneas que pueden estar en curso en un determinado instante de tiempo en esta red?

b. Suponga que todas las conexiones se encuentran entre el dispositivo de conmutación de la esquina superior izquierda y el dispositivo de conmutación de la esquina inferior derecha. ¿Cuál será el número máximo de conexiones que puede haber en curso?

P4. Repase la analogía de la caravana de coches de la Sección 1.4. Suponga una velocidad de propagación de 100 km/hora.

a. Suponga que la caravana se mueve a una velocidad de 150 km, partiendo de la caseta de peaje, pasando por una segunda caseta de peaje y deteniéndose justo después de la tercera caseta de peaje. ¿Cuál es el retardo terminal a terminal?

b. Repita el apartado (a) suponiendo ahora que en la caravana hay ocho coches en lugar de diez.

P5. En este problema se exploran los retardos de propagación y de transmisión, dos conceptos fundamentales en las redes de datos. Considere dos hosts, A y B, conectados por un enlace cuya velocidad es de R bps. Suponga que los dos hosts están separados m metros y que la velocidad de propagación a lo largo del enlace es igual a s metros/segundo. El host A envía un paquete de tamaño L bits al host B.

a. Exprese el retardo de propagación, d_{prop}, en función de m y s.

b. Determine el tiempo de transmisión del paquete, d_{trans}, en función de L y R.

c. Ignorando los retardos de procesamiento y de cola, obtenga una expresión para el retardo terminal a terminal.

d. Suponga que el host A comienza a transmitir el paquete en el instante $t = 0$. En el instante $t = d_{trans}$, ¿dónde estará el último bit del paquete?

e. Suponga que d_{prop} es mayor que d_{trans}. En el instante $t = d_{trans}$, ¿dónde estará el primer bit del paquete?

f. Suponga que d_{prop} es menor que d_{trans}. En el instante $t = d_{trans}$, ¿dónde estará el primer bit del paquete?

g. Suponga que $s = 2{,}5 \cdot 10^8$ metros/segundo, $L = 120$ bits y $R = 56$ kbps. Determine la distancia m de modo que d_{prop} sea igual a d_{trans}.

P6. En este problema vamos a considerar la transmisión de voz en tiempo real desde el host A al host B a través de una red de conmutación de paquetes (VoIP). El host A convierte sobre la marcha la voz analógica en un flujo de bits digital a 64 kbps. A continuación, el host A agrupa los bits en paquetes de 56 bytes. Entre el host A y el host B existe un enlace, cuya velocidad de transmisión es de 2 Mbps y su retardo de propaga-

ción es igual a 10 milisegundos. Tan pronto como el host A forma un paquete, lo envía al host B. Cuando el host B recibe un paquete completo, convierte los bits del paquete en una señal analógica. ¿Cuánto tiempo transcurre desde el momento en que se crea un bit (a partir de la señal analógica en el host A) hasta que se decodifica (como parte de la señal analógica en el host B)?

P7. Suponga que varios usuarios comparten un enlace de 3 Mbps. Suponga también que cada usuario requiere 150 kbps para transmitir y que sólo transmite durante el 10 por ciento del tiempo. (Véase la explicación sobre la multiplexación estadística de la Sección 1.3.)

a. Si se utiliza la conmutación de circuitos, ¿a cuántos usuarios puede darse soporte?

b. Para el resto de este problema, suponga que se utiliza una red de conmutación de paquetes. Halle la probabilidad de que un determinado usuario esté transmitiendo.

c. Suponga que hay 120 usuarios. Determine la probabilidad de que en un instante determinado haya exactamente n usuarios transmitiendo simultáneamente. (*Sugerencia*: utilice la distribución binomial.)

d. Calcule la probabilidad de que haya 21 o más usuarios transmitiendo simultáneamente.

P8. Consulte la explicación acerca de la multiplexación estadística de la Sección 1.3, en la que se proporciona un ejemplo con un enlace a 1 Mbps. Los usuarios están generando datos a una velocidad de 100 kbps cuando están ocupados, pero sólo lo están con una probabilidad de $p = 0{,}1$. Suponga que el enlace a 1 Mbps se sustituye por un enlace a 1 Gbps.

a. ¿Cuál es el valor de N, el número máximo de usuarios a los que se les puede dar soporte simultáneamente, cuando se emplea una red de conmutación de circuitos?

b. Considere ahora que se utiliza una red conmutación de paquetes y que el número de usuarios es M. Proporcione una fórmula (en función de p, M, N) para determinar la probabilidad de que más de N usuarios estén enviando datos.

P9. Considere un paquete de longitud L que tiene su origen en el sistema terminal A y que viaja a través de tres enlaces hasta un sistema terminal de destino. Estos tres enlaces están conectados mediante dos dispositivos de conmutación de paquetes. Sean d_i, s_i y R_i la longitud, la velocidad de propagación y la velocidad de transmisión del enlace i, para $i = 1, 2, 3$. El dispositivo de conmutación de paquetes retarda cada paquete d_{proc}. Suponiendo que no se produce retardo de cola, ¿cuál es el retardo total terminal a terminal del paquete en función de d_i, s_i, R_i, $(i = 1, 2, 3)$ y L? Suponga ahora que la longitud del paquete es de 1.500 bytes, la velocidad de propagación en ambos enlaces es igual a $2{,}5 \cdot 10^8$ m/s, la velocidad de transmisión en los tres enlaces es de 2 Mbps, el retardo de procesamiento en el conmutador de paquetes es de 3 milisegundos, la longitud del primer enlace es de 5.000 km, la del segundo de 4.000 km y la del último enlace es de 1.000 km. Para estos valores, ¿cuál es el retardo terminal a terminal?

P10. En el problema anterior, suponga que $R_1 = R_2 = R_3 = R$ y $d_{proc} = 0$. Suponga también que el conmutador de paquetes no almacena los paquetes y los reenvía, sino que transmite inmediatamente cada bit que recibe sin esperar a que llegue el paquete completo. ¿Cuál será el retardo terminal a terminal?

P11. Un conmutador de paquetes recibe un paquete y determina el enlace saliente por el que deberá ser reenviado. Cuando el paquete llega, hay otro paquete ya transmitido hasta la mitad por el mismo enlace de salida y además hay otros cuatro paquetes esperando para ser transmitidos. Los paquetes se transmiten según el orden de llegada. Suponga que todos los paquetes tienen una longitud de 1.500 bytes y que la velocidad del enlace es de 2 Mbps. ¿Cuál es el retardo de cola para el paquete? En sentido más general, ¿cuál es el retardo de cola cuando todos los paquetes tienen una longitud L, la velocidad de transmisión es R, x bits del paquete que se está transmitiendo actualmente ya han sido transmitidos y hay n paquetes en la cola esperando a ser transmitidos?

P12. Suponga que N paquetes llegan simultáneamente a un enlace en el que actualmente no se está transmitiendo ningún paquete ni tampoco hay ningún paquete en cola. Cada paquete tiene una longitud L y el enlace tiene una velocidad de transmisión R. ¿Cuál es el retardo medio de cola para los N paquetes?

P13. Considere el retardo de cola en el buffer de un router (que precede a un enlace de salida). Suponga que todos los paquetes tienen L bits, que la velocidad de transmisión es R bps y que llegan simultáneamente N paquetes al buffer cada LN/R segundos. Calcule el retardo medio de cola de un paquete. (*Sugerencia*: el retardo de cola para el primer paquete es igual a cero; para el segundo paquete es L/R; para el tercero es $2L/R$. El paquete N ya habrá sido transmitido cuando el segundo lote de paquetes llegue.)

P14. Considere el retardo de cola en el buffer de un router. Sea I la intensidad de tráfico; es decir, $I = La/R$. Suponga que el retardo de cola puede expresarse como $IL/R\ (1 - I)$ para $I < 1$.

a. Determine una fórmula para calcular el retardo total, es decir, el retardo de cola más el retardo de transmisión.

b. Dibuje el retardo total en función de L/R.

P15. Sea a la velocidad de llegada de los paquetes a un enlace en paquetes/segundo y sea μ la velocidad de transmisión del enlace en paquetes/segundo. Basándose en la fórmula del retardo total (es decir, el retardo de cola más el retardo de transmisión) obtenida en el problema anterior, deduzca una fórmula para el retardo total en función de a y μ.

P16. Considere el buffer de un router que precede a un enlace de salida. En este problema utilizaremos la fórmula de Little, una fórmula famosa en la teoría de colas. Sea N el número medio de paquetes que hay en el buffer más el paquete que está siendo transmitido. Sea a la velocidad a la que los paquetes llegan al enlace. Sea d el retardo medio total (es decir, el retardo de cola más el retardo de transmisión) experimentado por un paquete. La fórmula de Little es $N = a \cdot d$. Suponga que como media, el buffer contiene 10 paquetes y que el retardo medio de cola de un paquete es igual a 10 milisegundos. La velocidad de transmisión del enlace es igual a 100 paquetes/segundo. Utilizando la fórmula de Little, ¿cuál es la velocidad media de llegada de los paquetes suponiendo que no se produce pérdida de paquetes?

P17. a. Generalice la fórmula del retardo terminal a terminal dada en la Sección 1.4.3 para velocidades de procesamiento, velocidades de transmisión y retardos de propagación heterogéneos.

b. Repita el apartado (a), pero suponiendo ahora que existe un retardo medio de cola d_{cola} en cada nodo.

P18. Realice un trazado de la ruta (Traceroute) entre un origen y un destino situados en un mismo continente para tres horas del día diferentes.

a. Determine la media y la desviación estándar de los retardos de ida y vuelta para cada una de las horas.

b. Determine el número de routers existente en la ruta para cada una de las horas. ¿Ha variado la ruta para alguna de las horas?

c. Intente identificar el número de redes de ISP que los paquetes de Traceroute atraviesan desde el origen hasta el destino. Los routers con nombres similares y/o direcciones IP similares deben considerarse como parte del mismo ISP. En sus experimentos, ¿los retardos más largos se producen en las interfaces situadas entre proveedores ISP adyacentes?

d. Repita el apartado anterior para un origen y un destino situados en diferentes continentes. Compare los resultados para ubicaciones en un mismo continente y en distintos continentes.

P19. Considere el ejemplo sobre la tasa de transferencia correspondiente a la Figura 1.20(b). Suponga que hay M parejas cliente-servidor en lugar de 10. Sean R_s, R_c y R las velocidades de los enlaces de servidor, de los enlaces de cliente y del enlace de red. Suponga que todos los enlaces tienen la capacidad suficiente y que no existe otro tráfico en la red que el generado por las M parejas cliente-servidor. Deduzca una expresión general para la tasa de transferencia en función de R_s, R_c, R, y M.

P20. Considere la Figura 1.19(b). Suponga que existen M rutas entre el servidor y el cliente. No hay dos rutas que compartan ningún enlace. La ruta k ($k = 1, \ldots, M$) consta de N enlaces con velocidades de transmisión iguales a $R_1^k, R_2^k, \ldots, R_N^k$. Si el servidor sólo puede utilizar una ruta para enviar datos al cliente, ¿cuál será la máxima tasa de transferencia que puede alcanzar dicho servidor? Si el servidor puede emplear las M rutas para enviar datos, ¿cuál será la máxima tasa de transferencia que puede alcanzar el servidor?

P21. Considere la Figura 1.19(b). Suponga que cada enlace entre el servidor y el cliente tiene una probabilidad de pérdida de paquetes p y que las probabilidades de pérdida de paquetes de estos enlaces son independientes. ¿Cuál es la probabilidad de que un paquete (enviado por el servidor) sea recibido correctamente por el receptor? Si un paquete se pierde en el camino que va desde el servidor hasta el cliente, entonces el servidor volverá a transmitir el paquete. Como media, ¿cuántas veces tendrá que retransmitir el paquete el servidor para que el cliente lo reciba correctamente?

P22. Considere la Figura 1.19(a). Suponga que sabemos que el enlace cuello de botella a lo largo de la ruta entre el servidor y el cliente es el primer enlace, cuya velocidad es R_s bits/segundo. Suponga que envíamos un par de paquetes uno tras otro desde el servidor al cliente y que no hay más tráfico que ese en la ruta. Suponga que cada paquete tiene un tamaño de L bits y que ambos enlaces presentan el mismo retardo de propagación d_{prop}.

a. ¿Cuál es el periodo entre llegadas de paquetes al destino? Es decir, ¿cuánto tiempo transcurre desde que el último bit del primer paquete llega hasta que lo hace el último bit del segundo paquete?

b. Suponga ahora que el enlace cuello de botella es el segundo enlace (es decir, $R_c < R_s$). ¿Es posible que el segundo paquete tenga que esperar en la cola de entrada del

segundo enlace? Explique su respuesta. Suponga ahora que el servidor envía el segundo paquete T segundos después de enviar el primero. ¿Qué valor debe tener T para garantizar que el segundo paquete no tenga que esperar en la cola de entrada del segundo enlace? Explique su respuesta.

P23. Suponga que necesita enviar de forma urgente 40 terabytes de datos de Boston a Los Ángeles. Dispone de un enlace dedicado a 100 Mbps para transferencia de datos. ¿Qué preferiría, transmitir los datos a través del enlace o utilizar FedEx para hacer el envío por la noche? Explique su respuesta.

P24. Se tienen dos hosts, A y B, separados 20.000 kilómetros y conectados mediante un enlace directo con $R = 2$ Mbps. Suponga que la velocidad de propagación por el enlace es igual a 2,5 10^8 metros/segundo.

a. Calcule el producto ancho de banda-retardo, $R \quad d_{prop}$.

b. Se envía un archivo cuyo tamaño es de 800.000 bits desde el host A al host B. Suponga que el archivo se envía de forma continua como un mensaje de gran tamaño. ¿Cuál es el número máximo de bits que habrá en el enlace en un instante de tiempo determinado?

c. Haga una interpretación del producto ancho de banda-retardo.

d. ¿Cuál es el ancho (en metros) de un bit dentro del enlace? ¿Es más grande que un campo de fútbol?

e. Deduzca una expresión general para la anchura de un bit en función de la velocidad de propagación s, la velocidad de transmisión R y la longitud del enlace m.

P25. Continuando con el Problema P24, suponga que podemos modificar R. ¿Para qué valor de R es el ancho de un bit tan grande como la longitud del enlace?

P26. Considere el Problema P24 pero ahora para un enlace con $R = 1$ Gbps.

a. Calcule el producto ancho de banda-retardo, $R \quad d_{prop}$.

b. Considere el envío de un archivo de 800.000 bits desde el host A al host B. Suponga que el archivo se envía de forma continua como si fuera un mensaje de gran tamaño. ¿Cuál es el número máximo de bits que puede haber en el enlace en cualquier instante de tiempo dado?

c. ¿Cuál es el ancho (en metros) de un bit dentro del enlace?

P27. Haciendo referencia de nuevo al problema P24.

a. ¿Cuánto tiempo tarda en enviarse el archivo suponiendo que se envía de forma continua?

b. Suponga ahora que el archivo se divide en 20 paquetes conteniendo cada uno de ellos 40.000 bits. Suponga también que el receptor confirma la recepción de cada paquete y que el tiempo de transmisión de un paquete de confirmación es despreciable. Por último, suponga que el emisor no puede transmitir un paquete hasta que el anterior haya sido confirmado. ¿Cuánto tiempo se tardará en enviar el archivo?

c. Compare los resultados obtenidos en los apartados (a) y (b).

P28. Suponga que existe un enlace de microondas a 10 Mbps entre un satélite geoestacionario y su estación base en la Tierra. El satélite toma una fotografía digital por minuto y la envía a la estación base. La velocidad de propagación es 2,4 10^8 metros/segundo.

a. ¿Cuál es el retardo de propagación del enlace?

b. ¿Cuál es el producto ancho de banda-retardo, $R \cdot d_{prop}$?

c. Sea x el tamaño de la fotografía. ¿Cuál es el valor mínimo de x para que el enlace de microondas esté transmitiendo continuamente?

P29. Considere la analogía de la compañía aérea utilizada en la Sección 1.5 dedicada a las capas y la adición de cabeceras a las unidades de datos del protocolo a medida que fluyen en sentido descendente por la pila de protocolos. ¿Existe algún concepto equivalente a la información de cabecera que pueda añadirse a los pasajeros y al equipaje a medida que descienden por la pila de protocolos de la compañía aérea?

P30. En las redes de conmutación de paquetes modernas, el host de origen segmenta los mensajes largos de la capa de aplicación (por ejemplo, una imagen o un archivo de música) en paquetes más pequeños y los envía a la red. Después, el receptor ensambla los paquetes para formar el paquete original. Este proceso se conoce como *segmentación de mensajes*. La Figura 1.28 ilustra el transporte terminal a terminal de un mensaje con y sin segmentación del mensaje. Imagine que se envía un mensaje cuya longitud es de $8 \cdot 10^6$ bits desde el origen hasta el destino mostrados en la Figura 1.28. Suponga que cada enlace de los mostrados en la figura son enlaces a 2 Mbps. Ignore los retardos de propagación, de cola y de procesamiento.

a. Suponga que el mensaje se transmite desde el origen al destino *sin* segmentarlo. ¿Cuánto tiempo tarda el mensaje en desplazarse desde el origen hasta el primer conmutador de paquetes? Teniendo en cuenta que cada conmutador de paquetes utiliza el método de conmutación de almacenamiento y reenvío, ¿cuál el tiempo total que invierte el mensaje para ir desde el host de origen hasta el host de destino?

b. Suponga ahora que el mensaje se segmenta en 4.000 paquetes y que la longitud de cada paquete es de 2.000 bits. ¿Cuánto tiempo tarda el primer paquete en transmitirse desde el origen hasta el primer conmutador de paquetes? Cuando se está enviando el primer paquete del primer conmutador al segundo, el host de origen envía un segundo paquete al primer conmutador de paquetes. ¿En qué instante de tiempo habrá recibido el primer conmutador el segundo paquete completo?

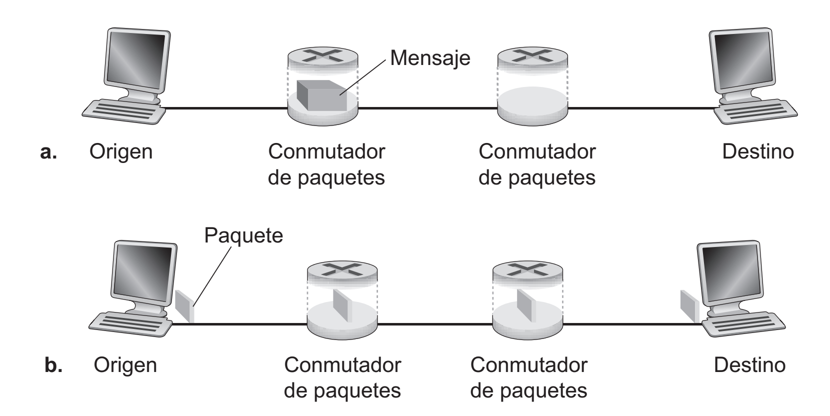


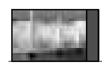
Figura 1.28 • Transporte de mensajes terminal a terminal: (a) sin segmentación de mensajes; (b) con segmentación de mensajes.

c. ¿Cuánto tiempo tarda en transmitirse el archivo desde el host de origen al host de destino cuando se emplea la segmentación de mensajes? Compare este resultado con la respuesta del apartado (a) y coméntelo.

d. Comente los inconvenientes de la segmentación de mensajes.

P31. Experimente con la applet *Message Segmentation* (segmentación de mensajes) disponible en el sitio web del libro. ¿Se corresponden los retardos indicados en el applet con los retardos del problema anterior? ¿Cómo afectan los retardos de propagación del enlace al retardo global terminal a terminal de la conmutación de paquetes (con segmentación de mensajes) y de la conmutación de mensajes?

P32. Se envía un archivo de gran tamaño de F bits desde el host A al host B. Entre los hosts A y B hay tres enlaces (y dos dispositivos de conmutación) y los enlaces no están congestionados (es decir, no existen retardos de cola). El host A divide el archivo en segmentos de S bits y añade 80 bits de cabecera a cada segmento, formando paquetes de $L = 80 + S$ bits. La velocidad de transmisión de cada enlace es de R bps. Calcule el valor de S que minimiza el retardo al transmitir el archivo desde el host A al host B. No tenga en cuenta el retardo de propagación.



Preguntas para la discusión

D1. ¿Qué tipos de servicios inalámbricos móviles hay disponibles en la región donde vive?

D2. Utilizando la tecnología LAN inalámbrica 802.11, diseñe una red doméstica para su casa o la de sus padres. Enumere los modelos específicos de los productos, así como los costes correspondientes de su red doméstica.

D3. Describa los servicios Skype de PC a PC. Pruebe el servicio de vídeo de Skype de PC a PC y redacte un informe narrando la experiencia.

D4. Skype ofrece un servicio que permite realizar una llamada telefónica desde un PC a un teléfono tradicional. Esto significa que la llamada de voz debe pasar por Internet y una red telefónica. Explique cómo puede hacerse esto.

D5. ¿Qué es un Servicio de mensajes cortos (SMS, *Short Message Service*)? ¿En qué países y continentes es popular este servicio? ¿Es posible enviar un mensaje SMS desde un sitio web a un teléfono móvil?

D6. ¿Qué es la transmisión de flujos de vídeo almacenado? ¿Cuáles son algunos de los sitios web más populares que suministran actualmente flujos de vídeo?

D7. ¿Qué son los flujos P2P de vídeo en directo? ¿Cuáles son algunos de los sitios web más populares que proporcionan actualmente este servicio?

D8. Localice cinco empresas que proporcionen servicios de compartición de archivos P2P. Para cada empresa, ¿con qué tipo de archivos trabajan, es decir, qué tipo de contenido gestionan?

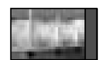
D9. ¿Quién inventó ICQ, es decir, el primer servicio de mensajería instantánea? ¿Cuándo fue inventado y cuál era la edad de sus inventores? ¿Quién inventó Napster, cuándo y qué edades tenían sus inventores?

D10. Compare el acceso a Internet inalámbrico WiFi y el acceso a Internet inalámbrico 3G. ¿Cuáles son las velocidades de bit de estos dos servicios? ¿Cuáles son los costes? Comente brevemente el concepto de itinerancia y ubicuidad de acceso.

D11. ¿Por qué ya no existe el servicio de compartición de archivos P2P de Napster original? ¿Qué es la RIAA y qué medidas se están tomando para limitar la compartición P2P de archivos con derechos de propiedad intelectual? ¿Cuál es la diferencia entre la infracción directa e indirecta de los derechos de propiedad intelectual?

D12. ¿Qué es BitTorrent? ¿Qué es lo que le hace fundamentalmente diferente de un servicio de compartición de archivos P2P como eDonkey, LimeWire o Kazaa?

D13. ¿Cree que dentro de 10 años seguirán compartiéndose de modo habitual archivos con derechos de propiedad intelectual a través de las redes de comunicaciones? ¿Por qué? Razone su respuesta.



Prácticas de laboratorio con Wireshark

"Dímelo y lo olvidaré. Enséñamelo y lo recordaré. Implícame y lo entenderé."

Proverbio chino

Para comprender mejor los protocolos de red se puede profundizar enormemente en ellos viéndolos en acción y observando, por ejemplo, la secuencia de mensajes intercambiados entre dos entidades, examinando los detalles de la operación del protocolo, haciendo que lleven a cabo determinadas acciones y observando dichas acciones y sus consecuencias. Esto puede hacerse en escenarios simulados o en un entorno de red real, como Internet. Los applets Java disponibles en el sitio web del libro aplican el primero de estos métodos. En el laboratorio con Wireshark se aplicará el segundo método. Se ejecutan aplicaciones de red en diversos escenarios utilizando una computadora doméstica, de una oficina o de un laboratorio de prácticas. Podrá observar los protocolos de red en su equipo, interactuando e intercambiando mensajes con entidades que se ejecutan en cualquier punto de Internet. Así, usted y su computadora serán una parte integral de estas prácticas de laboratorio. Podrá observar practicando y, de ese modo, aprender.

La herramienta básica para observar los mensajes intercambiados entre entidades que ejecutan protocolos es un *husmeador de paquetes* (***packet sniffer***). Como su nombre sugiere, un husmeador de paquetes copia de forma pasiva los mensajes que están siendo enviados y recibidos por una computadora; también muestra el contenido de los distintos campos de protocolo de los mensajes capturados. En la Figura 1.29 se muestra una captura de pantalla del software Wireshark. Wireshark es un husmeador de paquetes gratuito que se ejecuta en sistemas Windows, Linux/Unix y Mac. A lo largo del libro, encontrará prácticas de laboratorio con Wireshark que le permitirán explorar los protocolos estudiados en el capítulo. En la primera práctica de laboratorio con Wireshark, tendrá que conseguir e instalar una copia de Wireshark, acceder a un sitio web y capturar y examinar los mensajes de protocolo que estén siendo intercambiados entre su navegador web y el servidor web.

Puede encontrar todos los detalles acerca de esta primera práctica de laboratorio con Wireshark (incluyendo las instrucciones acerca de cómo obtener e instalar Wireshark) en el sitio web http://www.awl.com/kurose-ross.

Menú de **comandos**

Listado de paquetes capturados

Detalles de la cabecera del paquete seleccionado

Contenido del paquete en hexadecimal y ASCII

Figura 1.29 • Una captura de pantalla de Wireshark.

UNA ENTREVISTA CON...

Leonard Kleinrock



Leonard Kleinrock es catedrático de Ciencias de la Computación en la Universidad de California, Los Ángeles. En 1969, su computadora en UCLA se convirtió en el primer nodo de Internet. Su definición de los principios de la conmutación de paquetes en 1961 se convirtió en la tecnología en la que hoy se basa Internet. Recibió su licenciatura en el City College of New York (CCNY) y sus títulos de máster y doctor en Ingeniería Eléctrica en el MIT.

¿Qué le hizo decidirse a especializarse en la tecnología de redes e Internet?

Como estudiante de doctorado en el MIT en 1959, me di cuenta de que la mayor parte de mis compañeros de clase estaban realizando sus investigaciones en el área de la teoría de la información y de la teoría de la codificación. En el MIT se encontraba el gran investigador Claude Shannon, quien había abierto estos campos de investigación y que ya había resuelto la mayor parte de los problemas importantes. Los problemas de investigación que quedaban eran muy complicados y de consecuencias mucho menos importantes. Así que decidí iniciarme en una nueva área en la que nadie había pensado todavía. En el MIT, estaba rodeado de montones de computadoras y vi claramente que pronto esas máquinas necesitarían comunicarse entre sí. En aquel momento, no existía una forma efectiva de hacerlo, por lo que decidí desarrollar la tecnología que permitiría crear redes de datos eficientes y fiables.

¿Cuál fue su primer trabajo en la industria informática? ¿Qué significó para usted?

Entre 1951 y 1957 realicé en el CCNY los estudios de grado en Ingeniería Eléctrica en el turno de tarde. Durante el día, trabajé primero como técnico y luego como ingeniero en una pequeña empresa de electrónica industrial llamada Photobell. Mientras estuve allí, introduje la tecnología digital en sus líneas de productos. Básicamente, utilizábamos dispositivos fotoeléctricos para detectar la presencia de ciertos elementos (cajas, personas, etc.). El uso de un circuito que por entonces se denominaba *multivibrador biestable* era la clase de tecnología que necesitábamos para llevar el procesamiento digital al campo de la detección. Estos circuitos resultaron ser la base de las computadoras y ahora se conocen como *flip-flops, biestables* o *conmutadores* en la jerga actual.

¿Qué pasó por su cabeza cuando envió el primer mensaje de un host a otro (desde UCLA al Instituto de Investigación de Stanford)?

Francamente, no teníamos ni idea de la importancia de aquel suceso. No teníamos preparado un mensaje especial que pasara a la historia, como tantos otros inventores del pasado (Samuel Morse con "¡Lo que ha hecho Dios!", Alexander Graham Bell con "Watson, ¡ven aquí! Te necesito" o Neal Amstrong con "Un pequeño paso para el hombre, pero un gran paso para la Humanidad") ¡Aquellos tipos sí eran *inteligentes*! Conocían a los medios de comunicación y sabían lo que eran las relaciones públicas. Todo lo que nosotros queríamos hacer era iniciar una sesión en la computadora del SRI. Por lo que escribimos "L", lo que fue correctamente recibido, escribimos luego la letra "o", que fue recibida, y después la letra "g", que hizo que la computadora host del SRI fallara estrepitosamente. Así, nuestro mensaje fue el más corto de la historia: "Lo!".

Anteriormente, aquel año, yo había sido citado en una revista de UCLA diciendo que una vez que la red estuviera activa y funcionando, sería posible acceder a las utilidades informáticas desde nuestros

hogares y oficinas tan fácilmente como ya era disponer de electricidad y teléfono. Por tanto, mi visión en aquel momento era que Internet estaría en todas partes, siempre en funcionamiento, siempre disponible, que cualquiera con cualquier dispositivo podría conectarse desde cualquier lugar y que sería invisible. Sin embargo, nunca pude prever que mi madre con 99 años utilizaría Internet, y realmente la utilizó.

¿Cómo ve el futuro de las redes?

La parte fácil de la visión es predecir la propia infraestructura. Preveo que veremos una considerable implantación de la computación nómada, los dispositivos móviles y los espacios inteligentes. De hecho, la disponibilidad de dispositivos informáticos ligeros, baratos, de altas prestaciones y portátiles, y la disponibilidad de dispositivos de comunicaciones (combinado con la omnipresencia de Internet) nos ha permitido convertirnos en nómadas. La computación nómada hace referencia a la tecnología que permite a los usuarios finales ir de un lugar a otro obteniendo acceso a los servicios de Internet de forma transparente, independientemente de por dónde viajen e independientemente del dispositivo que utilicen. La parte más complicada de esta visión de futuro es predecir las aplicaciones y servicios, que siempre nos han sorprendido de forma increíble (correo electrónico, tecnologías de búsqueda, la world-wide-web, los blogs, las redes sociales, la generación y compartición por parte de los usuarios de música, fotografías y vídeos, etc.). Nos encontramos en el amanecer de una nueva era de aplicaciones móviles sorprendentes e innovadoras, que podremos utilizar con nuestros dispositivos de mano.

La siguiente ola nos permitirá pasar del mundo virtual del ciberespacio al mundo físico de los espacios inteligentes. Nuestros entornos (mesas, paredes, vehículos, relojes, etc.) cobrarán vida con la tecnología, mediante actuadores, sensores, lógica, procesamiento, almacenamiento, cámaras, micrófonos, altavoces, pantallas y comunicación. Esta tecnología integrada permitirá a nuestro entorno proporcionar los servicios IP que deseemos. Cuando accedamos a una habitación, la habitación sabrá que hemos entrado. Podremos comunicarnos con nuestro entorno de forma natural, hablando en nuestra lengua materna; nuestras solicitudes generarán respuestas que nos presentarán páginas web en pantallas de pared, en los cristales de las gafas, en forma de texto hablado, de hologramas, etc.

Mirando un poco más lejos, preveo un futuro en red que incluya los siguientes componentes adicionales fundamentales. Veo agentes software inteligentes por toda la red, cuya función será escarbar en los datos, actuar de acuerdo con ellos, detectar tendencias y llevar a cabo tareas de forma dinámica y adaptativa. Preveo que habrá una cantidad de tráfico de red considerablemente mayor, generada no tanto por los seres humanos, sino por estos dispositivos integrados y esos agentes software inteligentes. Preveo que habrá grandes conjuntos de sistemas dotados de auto-organización y encargados de controlar esa red tan enorme y tan rápida. Preveo que habrá enormes cantidades de información viajando instantáneamente a través de esa red y viéndose sometida en el camino a enormes cantidades de procesamiento y de filtrado. Internet será, esencialmente, un sistema nervioso global que llegará a todas partes. Preveo que sucederán todas estas cosas y muchas más a medida que nos vayamos adentrando en el siglo XXI.

¿Qué personas le han inspirado profesionalmente?

Con diferencia, el que más me ha inspirado ha sido Claude Shannon del MIT, un brillante investigador que tenía la capacidad de relacionar sus ideas matemáticas con el mundo físico de una forma extremadamente intuitiva. Estuvo en el tribunal de mi tesis doctoral.

¿Tiene algún consejo para los estudiantes que se inician ahora en el campo de las redes y de Internet?

Internet y todo lo que esa red hace posible constituye una nueva frontera de grandes dimensiones, llena de desafíos asombrosos. Hay grandes oportunidades para la innovación y no hay que sentirse restringido por la tecnología actual. Lo que hay que hacer es abrir la mente e imaginar cómo podrían ser las cosas, y luego hacer que eso suceda.



CAPÍTULO 2

La capa de aplicación

Las aplicaciones de red son la *razón de ser* de una red de computadoras (si no pudiéramos concebir ninguna aplicación útil, no existiría la necesidad de diseñar protocolos de red para darlas soporte). En los últimos 40 años, se han creado muchas aplicaciones de red ingeniosas y sorprendentes. Entre estas aplicaciones se incluyen las clásicas aplicaciones basadas en texto que se hicieron populares en las décadas de 1970 y 1980, como son el correo electrónico de texto, el acceso remoto a computadoras, la transferencia de archivos, los grupos de noticias y los chats de texto. Entre éstas, también se incluye la aplicación por excelencia de mediados de la década de 1990: la World Wide Web, acompañada de la navegación web, las búsquedas web y el comercio electrónico. Además, tenemos que citar las dos aplicaciones estrella aparecidas a finales del milenio: la mensajería instantánea con listas de contactos y la compartición de archivos P2P. Asimismo, también se incluyen muchas aplicaciones de éxito de audio y vídeo, como la telefonía por Internet, la compartición de vídeos y los flujos de vídeo, la radio por Internet y la televisión IP (IPTV). Además, el incremento en el uso del acceso residencial de banda ancha y la creciente omnipresencia del acceso inalámbrico constituyen la base para nuevas y excitantes aplicaciones que nos esperan en el futuro.

En este capítulo vamos a estudiar los aspectos conceptuales y de implementación de las aplicaciones de red. Comenzaremos definiendo los conceptos fundamentales relativos a la capa de aplicación, incluyendo los servicios de red requeridos por las aplicaciones, los clientes y servidores, los procesos y las interfaces de la capa de transporte. Examinaremos en detalle varias aplicaciones de red, como la Web, el correo electrónico, el sistema DNS, la distribución de archivos en redes entre pares (P2P, *Peer-to-Peer*) y la telefonía Internet P2P. A continuación, nos ocuparemos del desarrollo de las aplicaciones de red, tanto sobre TCP como UDP. En particular, estudiaremos las API de sockets y echaremos un vistazo a algunas aplicaciones cliente-servidor simples implementadas en Java. También proporcionaremos al final del capítulo varias divertidas e interesantes tareas de programación de sockets.

La capa de aplicación es un lugar particularmente bueno para comenzar el estudio de los protocolos, ya que es un terreno familiar. Habitualmente, empleamos muchas de las aplicaciones que se basan en los protocolos que vamos a estudiar. Nos dará una idea adecuada de qué son los protocolos y nos servirá para introducir muchas de las cuestiones que tendremos que volver a ver cuando estudiemos los protocolos de las capas de transporte, de red y de enlace.

2.1 Principios de las aplicaciones de red

Imagine que se le ha ocurrido una idea para desarrollar una nueva aplicación de red. Es posible que esa aplicación llegue a hacer un gran servicio a la Humanidad, o que simplemente le guste a su profesor, le haga ganar una fortuna o, únicamente, le divierta desarrollarla. Sea cual sea la motivación, a continuación vamos a ver cómo transformar la idea en una aplicación de red real.

Básicamente, el desarrollo de una aplicación de red implica escribir programas que se ejecuten en distintos sistemas terminales y que se comuniquen entre sí a través de la red. Por ejemplo, en la aplicación Web se emplean dos programas diferentes que se comunican entre sí: el navegador que se ejecuta en el host del usuario (una computadora de escritorio, un portátil, una PDA, un teléfono móvil, etc.) y el programa del servidor web que se ejecuta en el host servidor web. Otro ejemplo sería el caso de un sistema de compartición de archivos P2P en el que se emplea un programa en cada host que participa en la comunidad de compartición de archivos. En este caso, los programas instalados en los distintos hosts pueden ser similares o idénticos.

Por tanto, al desarrollar su nueva aplicación tendrá que escribir software que se ejecutará en varios sistemas. Este software podría escribirse en C, Java o Python. Una cuestión importante es que no es necesario escribir software que se ejecute en los dispositivos del núcleo de la red, como por ejemplo los routers o los switches de la capa de enlace. Incluso aunque deseara escribir software de aplicación para estos dispositivos del núcleo de la red, no podría hacerlo. Como hemos visto en el Capítulo 1 y se ilustra en la Figura 1.24, los dispositivos del núcleo de la red no operan en la capa de aplicación, sino en las capas inferiores, específicamente en la capa de red e inferiores. Este diseño básico (que confina el software de aplicación a los sistemas terminales), que se muestra en la Figura 2.1, ha facilitado el rápido desarrollo y la implementación de una gran cantidad de aplicaciones de red.

2.1.1 Arquitecturas de las aplicaciones de red

Antes de profundizar en la codificación del software, deberíamos disponer de una visión general de la arquitectura de la aplicación. Tenga en cuenta que la arquitectura de una aplicación es muy distinta de la arquitectura de la red (como por ejemplo la arquitectura de Internet de cinco capas vista en el Capítulo 1). Desde la perspectiva del desarrollador de aplicaciones, la arquitectura de la red es fija y proporciona un conjunto específico de servicios a las aplicaciones. Por otro lado, el desarrollador de aplicaciones diseña la **arquitectura de la aplicación**, que establece cómo la aplicación debe estructurarse en los distintos sistemas terminales. Al seleccionar la arquitectura de la aplicación, el desarrollador probablemente utilizará uno de los dos paradigmas arquitectónicos predominantes en las aplicaciones de red modernas: la arquitectura cliente-servidor o la arquitectura P2P.

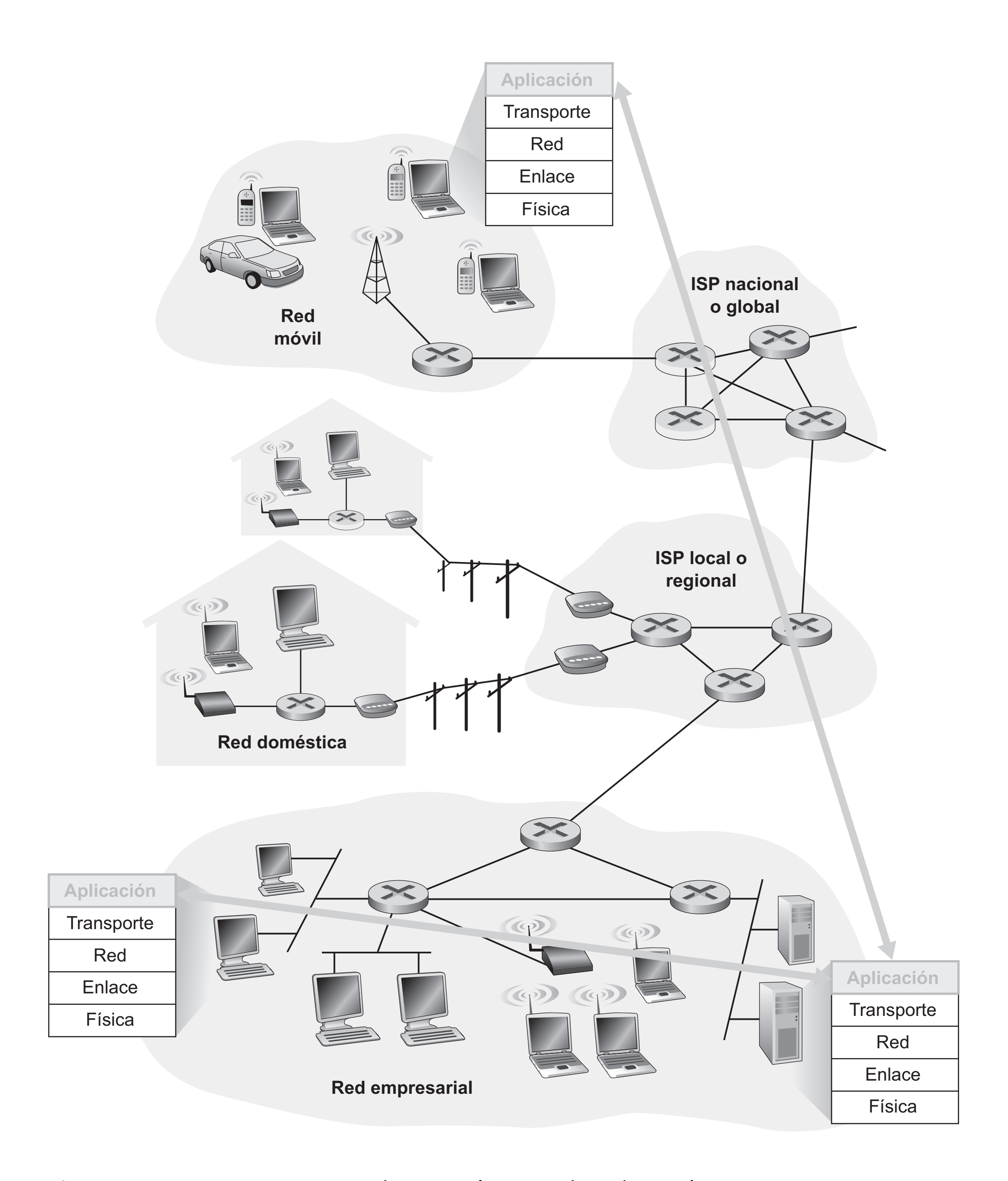


Figura 2.1 • La comunicación de una aplicación de red tiene lugar entre sistemas terminales en la capa de aplicación.

En una **arquitectura cliente-servidor** siempre existe un host activo, denominado *servidor*, que da servicio a las solicitudes de muchos otros hosts, que son los *clientes*. Los hosts clientes pueden estar activos siempre o de forma intermitente. Un ejemplo clásico es

la Web en la que un servidor web siempre activo sirve las solicitudes de los navegadores que se ejecutan en los hosts clientes. Cuando un servidor web recibe una solicitud de un objeto de un host cliente, responde enviándole el objeto solicitado. Observe que, con la arquitectura cliente-servidor, los clientes no se comunican directamente entre sí; por ejemplo, en la aplicación web, dos navegadores no se comunican entre sí. Otra característica de la arquitectura cliente-servidor es que el servidor tiene una dirección fija y conocida, denominada dirección IP (de la que hablaremos enseguida). Puesto que el servidor tiene una dirección fija y conocida, y siempre está activo, un cliente siempre puede contactar con él enviando un paquete a su dirección. Entre las aplicaciones más conocidas que utilizan la arquitectura cliente-servidor se encuentran las aplicaciones web, FTP, Telnet y de correo electrónico. En la Figura 2.2(a) se muestra la arquitectura cliente-servidor.

Normalmente, en una aplicación cliente-servidor un único host servidor es incapaz de responder a todas las solicitudes de sus clientes. Por ejemplo, el sitio de una red social popular puede verse rápidamente desbordado si sólo dispone de un servidor para gestionar todas las solicitudes. Por esta razón, en las arquitecturas cliente-servidor suele utilizarse una agrupación (*cluster*) de hosts, que a veces se denomina **centro de datos**, para crear un servidor virtual de gran capacidad. Los servicios de aplicaciones basadas en una arquitectura cliente-servidor a menudo precisan una **infraestructura intensiva**, ya que requieren que los proveedores de servicios compren, instalen y mantengan granjas de servidores. Además, los

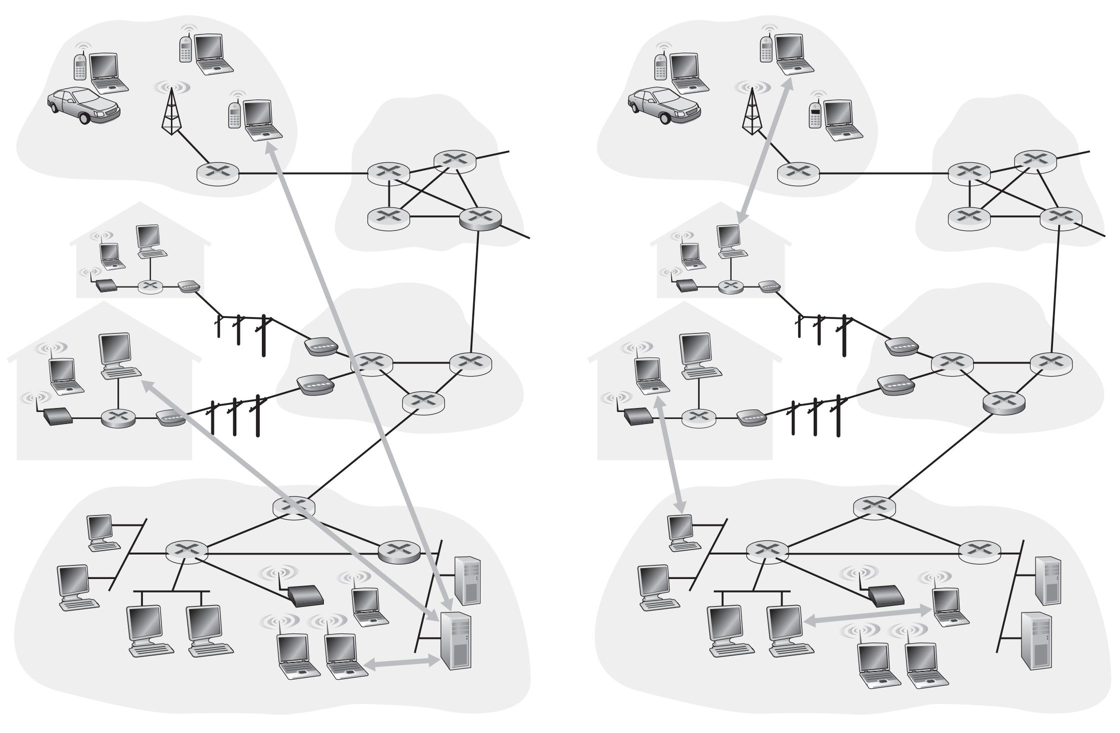


Figura 2.2 • (a) Arquitectura cliente-servidor; (b) arquitectura P2P.

proveedores del servicio deben afrontar los costes de las recurrentes interconexiones y ancho de banda para enviar datos a Internet y recibirlos. Servicios populares como los motores de búsqueda (por ejemplo, Google), el comercio por Internet (como Amazon y e-Bay), el correo electrónico web (como por ejemplo Yahoo Mail), las redes sociales (como MySpace y Facebook) y la compartición de vídeos (como YouTube) precisan una infraestructura intensiva y su suministro es enormemente costoso.

En una **arquitectura P2P** existe una mínima (o ninguna) dependencia de una infraestructura de servidores siempre activos. En su lugar, la aplicación explota la comunicación directa entre parejas de hosts conectados de forma intermitente, conocidos como *peers* (pares). Los pares no son propiedad del proveedor del servicio, sino que son las computadoras de escritorio y portátiles controlados por usuarios, encontrándose la mayoría de los pares en domicilios, universidades y oficinas. Puesto que los pares se comunican sin pasar por un servidor dedicado, la arquitectura se denomina arquitectura *peer-to-peer* (P2P). Muchas de las aplicaciones actuales más populares y con un elevado nivel de tráfico están basadas en arquitecturas P2P. Entre estas aplicaciones se incluyen la distribución de archivos (por ejemplo, BitTorrent), la compartición de archivos (como eMule y LimeWire), la telefonía por Internet (como Skype) e IPTV (como PPLive). En la Figura 2.2(b) se ilustra la arquitectura P2P. Hemos mencionado que algunas aplicaciones tienen arquitecturas híbridas que combinan elementos cliente-servidor y P2P. Por ejemplo, en muchas aplicaciones de mensajería instantánea los servidores se utilizan para hacer un seguimiento de las direcciones IP de los usuarios, pero los mensajes de usuario a usuario se envían directamente entre los hosts de dichos usuarios (sin pasar por los servidores intermedios).

Una de las características más convincentes de las arquitecturas P2P es su **auto-escalabilidad**. Por ejemplo, en una aplicación de compartición de archivos P2P, aunque cada peer genera una carga de trabajo solicitando archivos, también añade capacidad de servicio al sistema distribuyendo archivos a otros peers. Las arquitecturas P2P también presentan una buena relación coste-prestaciones, ya que normalmente no requieren una infraestructura de servidores significativa ni un gran ancho de banda de servidor. Para reducir costes, los proveedores de servicios (MSN, Yahoo, etc.) cada vez están más interesados en emplear las arquitecturas P2P para sus aplicaciones [Chuang 2007]. Sin embargo, las futuras aplicaciones P2P se enfrentan a tres retos importantes:

1. *Orientadas al ISP.* La mayoría de los ISP residenciales (incluyendo los ISP de líneas DSL y cable) están dimensionados para hacer un uso “asimétrico” del ancho de banda, es decir, para dar soporte a mucho más tráfico de descarga que de carga. Pero las aplicaciones P2P para distribución de archivos y de flujos de vídeo desplazan el tráfico de carga de los servidores a los ISP residenciales, ejerciendo en consecuencia una gran presión sobre los ISP. Las aplicaciones P2P futuras tendrán que ser diseñadas pensando en los ISP [Xie 2008].
2. *Seguridad.* Debido a su naturaleza extremadamente distribuida y abierta, las aplicaciones P2P pueden ser un reto para la seguridad [Doucer 2002; Yu 2006; Liang 2006; Naoumov 2006; Dhungel 2008].
3. *Incentivos.* El éxito de las aplicaciones P2P futuras también depende de convencer a los usuarios para ofrecer voluntariamente a las aplicaciones recursos de ancho de banda, de almacenamiento y de computación, lo que constituye todo un reto de diseño de incentivos [Feldman 2005; Piatek 2008; Aperjis 2008].

2.1.2 Procesos de comunicación

Antes de crear su aplicación de red, también necesita disponer de unos conocimientos básicos sobre cómo los programas que se ejecutan en varios sistemas terminales se comunican entre sí. En la jerga de los sistemas operativos, realmente no son los programas sino los **procesos** los que se comunican. Un proceso puede interpretarse como un programa que se ejecuta dentro de un sistema terminal. Cuando los procesos se ejecutan en el mismo sistemas terminal, pueden comunicarse entre sí mediante la comunicación entre procesos aplicando las reglas gobernadas por el sistema operativo del sistema terminal. Pero, en este libro, no estamos especialmente interesados en cómo se comunican los procesos que tienen lugar en un mismo host, sino en cómo se comunican los procesos que se ejecutan en hosts *diferentes* (con sistemas operativos potencialmente diferentes).

Los procesos de dos sistemas terminales diferentes se comunican entre ellos intercambiando **mensajes** a través de la red de computadoras. Un proceso emisor crea y envía mensajes a la red; un proceso receptor recibe estos mensajes y posiblemente responde devolviendo mensajes. La Figura 2.1 ilustra que los procesos se comunican entre sí utilizando la capa de aplicación de la pila de protocolos de cinco capas.

Procesos cliente y servidor

Una aplicación de red consta de parejas de procesos que se envían mensajes entre sí a través de una red. Por ejemplo, en la aplicación web, un proceso de un navegador cliente intercambia mensajes con un proceso de un servidor web. En un sistema de compartición de archivos P2P, se transfiere un archivo desde un proceso de un peer a un proceso de otro par. Normalmente, en una pareja de procesos que están comunicándose, etiquetamos a uno de los procesos como el **cliente** y al otro como el **servidor**. En una aplicación web, un navegador es un proceso cliente y el servidor web es un proceso servidor. En la compartición de archivos P2P, el host que descarga el archivo se etiqueta como el cliente y el host que está cargando el archivo se etiqueta como el servidor.

Es posible que haya observado que en algunas aplicaciones, tales como la compartición de archivos P2P, un proceso puede ser tanto un cliente como un servidor. De hecho, un proceso en un sistema de compartición de archivos P2P puede cargar y descargar archivos. No obstante, en el contexto de cualquier sesión de comunicación entre una pareja de procesos, podemos etiquetar a uno de los procesos como el cliente y al otro como el servidor. Definimos los procesos cliente y servidor como sigue:

> *En el contexto de una sesión de comunicación entre una pareja de procesos, el proceso que inicia la comunicación (es decir, que inicialmente se pone en contacto con el otro proceso al principio de la sesión) se etiqueta como el **cliente**. El proceso que espera a ser contactado para comenzar la sesión es el **servidor**.*

En la Web, un proceso de navegador inicia el contacto con un proceso de servidor web; por tanto, el proceso de navegador es el cliente y el proceso de servidor web es el servidor. En la compartición de archivos P2P, cuando un par A pide a un par B que le envíe un determinado archivo, el A es el cliente y el B es el servidor en el contexto de esta sesión de comunicación concreta. Si no existe ningún tipo de confusión, en ocasiones, también emplearemos la terminología "lado del cliente y lado del servidor de una aplicación". Al final del capítulo, examinaremos un código simple tanto para el lado del cliente como para el lado del servidor de las aplicaciones de red.

Interfaz entre el proceso y la red de computadoras

Como hemos mencionado anteriormente, la mayoría de las aplicaciones constan de parejas de procesos de comunicación que se envían mensajes entre ellos. Cualquier mensaje enviado de un proceso al otro debe atravesar la red subyacente. Un proceso envía mensajes a la red y los recibe de la red a través de una interfaz software denominada ***socket***. Veamos una analogía que nos ayudará a comprender los conceptos de proceso y socket. Un proceso es análogo a una casa y un socket es análogo a la puerta de la casa. Cuando un proceso desea enviar un mensaje a otro proceso que se está ejecutando en otro host, envía el mensaje a través de la puerta (socket). Este proceso emisor supone que existe una infraestructura de transporte al otro lado de la puerta que llevará el mensaje hasta la puerta del proceso de destino. Una vez que el mensaje llega al host de destino, éste pasa a través de la puerta (socket) del proceso receptor, actuando entonces el proceso receptor sobre el mensaje.

La Figura 2.3 ilustra la comunicación mediante sockets entre dos procesos que se comunican a través de Internet. (En la Figura 2.3 se supone que el protocolo de transporte subyacente utilizado por los procesos es el protocolo TCP de Internet.) Como se muestra en la figura, un socket es la interfaz entre la capa de aplicación y la capa de transporte de un host. También se conoce como **Interfaz de programación de aplicaciones (API, *Application Programming Interface*)** que opera entre la aplicación y la red, ya que el socket es la interfaz de programación con la que se construyen las aplicaciones de red. El desarrollador de la aplicación tiene el control sobre todos los elementos del lado de la capa de aplicación del socket pero apenas tiene control sobre el lado de la capa de transporte del socket. El único control que tiene el desarrollador de la aplicación sobre el lado de la capa de transporte es (1) la elección del protocolo de transporte y (2) quizá la capacidad de fijar unos pocos parámetros de la capa de transporte, como por ejemplo los tamaños máximo del buffer y de segmento (lo que veremos en el Capítulo 3). Una vez que el desarrollador de la aplicación ha seleccionado un protocolo de transporte (si hay disponibles varios entre los que elegir), la aplicación se crea utilizando los servicios de la capa de transporte proporcionados por dicho protocolo. En las Secciones 2.7 y 2.8 exploraremos en detalle los sockets.

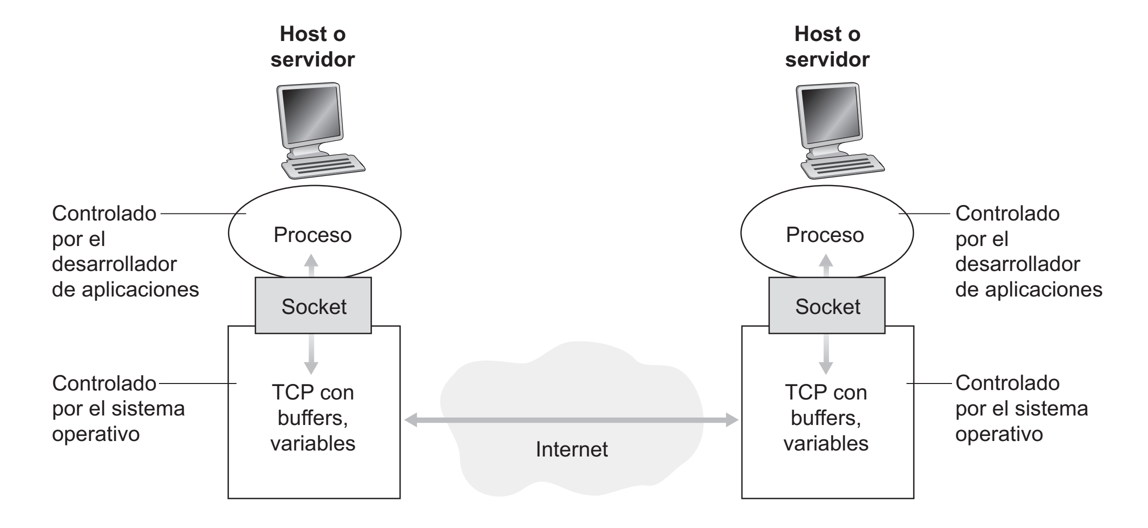


Figura 2.3 • Procesos de aplicación, sockets y protocolo de transporte subyacente.

2.1.3 Servicios de transporte disponibles para las aplicaciones

Recordemos que un socket es la interfaz entre el proceso de la aplicación y el protocolo de la capa de transporte. La aplicación del lado emisor empuja los mensajes a través del socket. En el otro lado del socket, el protocolo de la capa de transporte tiene la responsabilidad de llevar los mensajes hasta la "puerta" del socket de recepción.

Muchas redes, incluyendo Internet, proporcionan más de un protocolo de la capa de transporte. Cuando vaya a desarrollar una aplicación, tendrá que elegir uno de los protocolos de la capa de transporte disponibles. ¿Cómo llevar a cabo esta selección? Muy probablemente, tendrá que estudiar los servicios que ofrecen los protocolos de la capa de transporte disponibles y después elegir aquel protocolo que proporcione los servicios que mejor se adapten a las necesidades de la aplicación. La situación es similar a tener que elegir entre viajar en tren o en avión para ir de una ciudad a otra. Tiene que elegir un medio de transporte u otro, y cada uno de ellos ofrece servicios diferentes. Por ejemplo, el tren le ofrece partir y llegar al centro de las ciudades, mientras que el avión ofrece un tiempo de viaje más corto.

¿Cuáles son los servicios que puede ofrecer un protocolo de la capa de transporte a las aplicaciones que le invocan? Podemos clasificar los posibles servicios de forma bastante general según cuatro parámetros: transferencia de datos fiable, tasa de transferencia, temporización y seguridad.

Transferencia de datos fiable

Como se ha explicado en el Capítulo 1, en una red de computadoras pueden perderse paquetes. Por ejemplo, un paquete puede desbordar el buffer de un router, o podría ser descartado por un host o un router después de comprobar que algunos de sus bits están corrompidos. En muchas aplicaciones (como el correo electrónico, la transferencia de archivos, el acceso remoto a hosts, la transferencia de documentos web y las aplicaciones financieras) la pérdida de datos puede tener consecuencias catastróficas (en el último caso, para el banco y para ¡el cliente!). Por tanto, para dar soporte a estas aplicaciones, es preciso hacer algo para garantizar que los datos enviados desde un terminal de la aplicación sean todos ellos entregados correcta y completamente al otro terminal de la aplicación. Si un protocolo proporciona un servicio de entrega de datos garantizado, se dice que proporciona una **transferencia de datos fiable**. Un servicio importante que un protocolo de la capa de transporte puede potencialmente proporcionar a una aplicación es la transferencia de datos fiable proceso a proceso. Cuando un protocolo de transporte suministra este servicio, el proceso emisor puede pasar sus datos al socket y sabe con certidumbre absoluta que los datos llegarán sin errores al proceso receptor.

Si un protocolo de la capa de transporte no proporciona una transferencia de datos fiable, los datos enviados por el proceso emisor pueden no llegar nunca al proceso de recepción. Esto puede ser aceptable para **aplicaciones tolerantes a pérdidas**; por ejemplo, la mayor parte de las aplicaciones multimedia como las de audio/vídeo en tiempo real o las de audio/vídeo almacenado pueden tolerar que cierta cantidad de datos se pierda. En estas aplicaciones multimedia, la pérdida de datos puede dar lugar a una pequeña interrupción al reproducir el audio/vídeo, lo que no constituye un problema fundamental.

Tasa de transferencia

En el Capítulo 1 hemos presentado el concepto de tasa de transferencia disponible, el cual podemos trasladar al contexto de una sesión de comunicaciones entre dos procesos a lo largo de una ruta de red como la tasa a la que el proceso emisor puede suministrar bits al proceso de recepción. Puesto que otras sesiones compartirán el ancho de banda a lo largo de la ruta de red y puesto que esas otras sesiones se iniciarán y terminarán aleatoriamente, la tasa de transferencia disponible puede fluctuar con el tiempo. Estas observaciones nos llevan a otro servicio que un protocolo de la capa de transporte podría proporcionar, una tasa de transferencia disponible garantizada a un cierta velocidad especificada. Con un servicio así, la aplicación podría solicitar una tasa de transferencia garantizada de *r* bits/segundo y el protocolo de transporte podría entonces garantizar que la tasa de transferencia disponible sea siempre al menos de *r* bits/segundo. Un servicio que ofreciera una tasa de transferencia garantizada resultaría atractivo para muchas aplicaciones. Por ejemplo, si una aplicación de telefonía por Internet codifica voz a 32 kbps, tendrá que enviar datos a la red y tendrá que entregar los datos a la aplicación receptora a esa velocidad. Si el protocolo de transporte no puede proporcionar esa tasa de transferencia, la aplicación tendrá que realizar la codificación a una velocidad menor (y recibir a una tasa de transferencia adecuada como para mantener esa velocidad de codificación más lenta) o bien tendrá que renunciar, puesto que recibir a la mitad de la tasa de transferencia necesaria no tiene ninguna utilidad para esta aplicación de telefonía por Internet. Las aplicaciones con requisitos de tasa de transferencia se conocen como **aplicaciones sensibles al ancho de banda**. Muchas aplicaciones multimedia actuales son sensibles al ancho de banda, pero algunas de ellas pueden emplear técnicas de codificación adaptativa para realizar la codificación a una velocidad que se adapte a la tasa de transferencia disponible actualmente.

Mientras que las aplicaciones sensibles al ancho de banda tienen que cumplir requisitos específicos para la tasa de transferencia, las **aplicaciones elásticas** pueden hacer uso de la tasa de transferencia, mucha o poca, que haya disponible. El correo electrónico, la transferencia de archivos y las transferencias web son todas ellas aplicaciones elásticas. Por supuesto, cuanto mayor sea la tasa de transferencia, mejor.

Temporización

Un protocolo de la capa de transporte también puede proporcionar garantías de temporización. Al igual que con las tasas de transferencia garantizadas, las garantías de temporización también pueden darse de diversas formas. Un ejemplo de garantía podría ser que cada bit que el emisor empuja por su socket llegue al socket del receptor en no más de 100 milisegundos. Un servicio así sería atractivo para las aplicaciones interactivas en tiempo real, como la telefonía por Internet, los entornos virtuales, la teleconferencia y los juegos multijugador, todas las cuales requieren restricciones de temporización muy estrictas sobre la entrega de datos para ser efectivas. (Véase el Capítulo 7 y [Gauthier 1999; Ramjee 1994].) Por ejemplo, los retardos largos en la telefonía por Internet tienden a dar lugar a pausas antinaturales en una conversación; en un juego multijugador o en un entorno virtual interactivo, un retardo largo entre la realización de una acción y la visualización de la respuesta del entorno (por ejemplo, de otro jugador que se encuentra en el otro extremo de una conexión extremo a extremo) hace que la aplicación parezca menos realista. En las aplicaciones que no se ejecutan en tiempo real, siempre es preferible un retardo pequeño que grande, pero no se aplican restricciones estrictas a los retardos extremo a extremo.

Seguridad

Por último, un protocolo de transporte puede proporcionar a una aplicación uno o más servicios de seguridad. Por ejemplo, en el host emisor, un protocolo de transporte puede cifrar todos los datos transmitidos por el proceso emisor, y en el host receptor puede descifrar los datos antes de entregarlos al proceso receptor. Un servicio así debe proporcionar confidencialidad entre los dos procesos, incluso aunque los datos sean observados de alguna manera entre los procesos emisor y receptor. Un protocolo de transporte también puede proporcionar otros servicios de seguridad además del de la confidencialidad, como pueden ser los mecanismos para garantizar la integridad de los datos y mecanismos de autenticación en el punto terminal, temas que abordaremos en detalle en el Capítulo 8.

2.1.4 Servicios de transporte proporcionados por Internet

Hasta el momento hemos considerado los servicios de transporte que una red de computadoras *podría* proporcionar en general. Seamos ahora un poco más específicos y examinemos el tipo de soporte que Internet proporciona a las aplicaciones. Internet (y, de forma más general, las redes TCP/IP) pone a disposición de las aplicaciones dos protocolos de transporte: UDP y TCP. Cuando como desarrollador de aplicaciones cree una nueva aplicación de red para Internet, una de las primeras decisiones que tendrá que tomar es si utilizar UDP o TCP. Cada uno de estos protocolos ofrece un conjunto diferente de servicios a las aplicaciones que los invocan. La Figura 2.4 detalla los requisitos de servicio para algunas aplicaciones seleccionadas.

Servicios TCP

El modelo de servicio TCP incluye un servicio orientado a la conexión y un servicio de transferencia de datos fiable. Cuando una aplicación invoca TCP como su protocolo de transporte, la aplicación recibe estos dos servicios de TCP.

Aplicación	Pérdida de datos	Ancho de banda	Sensible al tiempo
Transferencia de archivos	Sin pérdidas	Elástica	No
Corre electrónico	Sin pérdidas	Elástica	No
Documentos web	Sin pérdidas	Elástica (pocos kbps)	No
Telefonía por Internet/ Videoconferencia	Tolerante a las pérdidas	Audio: unos pocos kbps–1 Mbps Vídeo: 10 kbps–5 Mbps	Sí: décimas de segundo
Audio/vídeo almacenado	Tolerante a las pérdidas	Como el anterior	Sí: unos pocos segundos
Juegos interactivos	Tolerante a las pérdidas	Unos pocos kbps–10 kbps	Sí: décimas de segundos
Mensajería instantánea	Sin pérdidas	Elástica	Sí y no

Figura 2.4 • Requisitos de algunas aplicaciones de red seleccionadas.

- *Servicio orientado a la conexión.* TCP hace que el cliente y el servidor intercambien la información de control de la capa de transporte entre sí *antes* de que empiecen a fluir los mensajes del nivel de aplicación. Este procedimiento denominado de negociación, de reconocimiento o de establecimiento de la conexión alerta al cliente y al servidor, permitiéndoles prepararse para el intercambio de paquetes. Después de esta fase de negociación, se dice que existe una **conexión TCP** entre los sockets de los dos procesos. La conexión es una conexión full-duplex ya que los dos procesos pueden enviarse mensajes entre sí a través de la conexión al mismo tiempo. Una vez que la aplicación ha terminado de enviar mensajes, es necesario desactivar la conexión. Se dice que es un servicio "orientado a la conexión" en lugar de un servicio de "conexión" porque los dos procesos están conectados de una forma muy laxa. En el Capítulo 3 examinaremos los servicios orientados a la conexión y veremos cómo se implementan.
- *Servicio de transferencia de datos fiable.* Los procesos de comunicación pueden confiar en TCP para entregar todos los datos enviados sin errores y en el orden correcto. Cuando un lado de la aplicación pasa un flujo de bytes a un socket, puede contar con TCP para entregar el mismo flujo de bytes al socket receptor sin pérdida ni duplicación de bytes.

TCP también incluye un mecanismo de control de congestión, que es un servicio para mejorar el funcionamiento general de Internet, más que para el beneficio directo de los procesos que se comunican. Este mecanismo de control de congestión de TCP regula el proceso emisor (cliente o servidor) cuando la red está congestionada entre el emisor y el receptor. Como se explica en el Capítulo 3, el control de congestión de TCP también intenta limitar cada conexión TCP para que utilice una cuota equitativa de ancho de banda de la red. La regulación de la velocidad de transmisión puede tener efectos muy dañinos sobre las aplicaciones de audio y de vídeo en tiempo real, que tienen unos requisitos mínimos de tasa de transferencia. Además, las aplicaciones en tiempo real son tolerantes a las pérdidas y no necesitan un servicio de transporte completamente fiable. Por estas razones, los desarrolladores de aplicaciones en tiempo real a menudo deciden ejecutar sus aplicaciones utilizando el protocolo UDP en lugar de TCP.

Servicios UDP

UDP es un protocolo de transporte ligero simple que proporciona unos servicios mínimos y no está orientado a la conexión, por lo que no tiene lugar un procedimiento de negociación antes de que los dos procesos comiencen a comunicarse. UDP proporciona un servicio de transferencia de datos no fiable; es decir, cuando un proceso envía un mensaje a un socket UDP, el protocolo UDP no ofrece *ninguna* garantía de que el mensaje vaya a llegar al proceso receptor. Además, los mensajes que sí llegan al proceso receptor pueden hacerlo de manera desordenada.

UDP no incluye tampoco un mecanismo de control de congestión, por lo que el lado emisor de UDP puede introducir datos en la capa inferior (la capa de red) a la velocidad que le parezca. (Sin embargo, debe observar que la tasa de transferencia extremo a extremo real puede ser menor que esta velocidad a causa del ancho de banda limitado de los enlaces intervinientes o a causa de la congestión.) Puesto que las aplicaciones en tiempo real a menudo pueden tolerar ciertas pérdidas pero requieren una velocidad mínima para ser efectivas, los desarrolladores de estas aplicaciones en ocasiones deciden ejecutarlas usando UDP, soslayando los mecanismos de control de congestión y la sobrecarga de gestión de los paquetes TCP. Por otro lado, dado que muchos cortafuegos están configurados para bloquear casi



SEGURIDAD

TCP SEGURO

Ni TCP ni UDP proporcionan ningún mecanismo de cifrado (los datos que el proceso emisor pasa al socket son los mismos datos que viajan a través de la red hasta el proceso de destino). Luego, por ejemplo, si el proceso emisor envía una contraseña en texto legible (es decir, no cifrada) a su socket, esa contraseña viajará a través de todos los enlaces entre el emisor y el receptor, pudiendo ser husmeada y descubierta en cualquiera de los enlaces intervinientes. Puesto que la confidencialidad y otras cuestiones de seguridad son críticas para muchas aplicaciones, la comunidad de Internet ha desarrollado una mejora para TCP, denominada **SSL** (*Secure Sockets Layer*, Capa de conectores seguros). TCP mejorado con SSL no sólo hace todo lo que hace el protocolo TCP tradicional, sino que también proporciona servicios críticos de seguridad proceso a proceso, entre los que se incluyen mecanismos de cifrado, de integridad de los datos y de autenticación en el punto terminal. Debemos destacar que SSL no es un tercer protocolo de transporte de Internet, al mismo nivel que TCP y UDP, sino que es una mejora de TCP, que se implementa en la capa de aplicación. En concreto, si una aplicación desea utilizar los servicios de SSL, tiene que incluir código SSL (existen clases y librerías enormemente optimizadas) tanto en el lado del cliente como en el del servidor de la aplicación. SSL tiene su propia API de sockets, que es similar a la API de sockets del protocolo TCP tradicional. Cuando una aplicación utiliza SSL, el proceso emisor pasa los datos en texto legible al socket SSL; a continuación, SSL cifra los datos en el host emisor y los pasa al socket TCP. Los datos cifrados viajan a través de Internet hasta el socket TCP del proceso receptor. El socket de recepción pasa los datos cifrados a SSL, que los descifra. Por último, SSL pasa los datos en texto legible a través de su socket al proceso receptor. En el Capítulo 8 se cubre en detalle SSL.

todos los tipos de tráfico UDP, los diseñadores están decidiendo cada vez más frecuentemente ejecutar las aplicaciones multimedia en tiempo real sobre TCP [Sripanidkulchai 2004].

Servicios no proporcionados por los protocolos de transporte de Internet

Hemos organizado los posibles servicios del protocolo de transporte según cuatro parámetros: transferencia de datos fiable, tasa de transferencia, temporización y seguridad. ¿Cuáles de estos servicios proporcionan TCP y UDP? Ya hemos mencionado que TCP proporciona transferencia de datos extremo a extremo fiable. Y también sabemos que TCP se puede mejorar fácilmente en la capa de aplicación con SSL para proporcionar servicios de seguridad. Pero en esta breve descripción de TCP y UDP hemos omitido notoriamente hacer mención de las garantías relativas a la tasa de transferencia o la temporización (servicios que *no* proporcionan los protocolos de transporte de Internet de hoy día). ¿Significa esto que las aplicaciones sensibles al tiempo como la telefonía por Internet no se pueden ejecutar actualmente en Internet? Evidentemente, la respuesta es no, Internet lleva muchos años albergando aplicaciones sensibles al tiempo. Estas aplicaciones suelen funcionar bastante bien porque han sido diseñadas para hacer frente a esta falta de garantías de la mejor forma posible. En

Aplicación	Protocolo de la capa de aplicación	Protocolo de transporte subyacente
Correo electrónico	SMTP [RFC 5321]	TCP
Acceso remoto a terminal	Telnet [RFC 854]	TCP
Web	HTTP [RFC 2616]	TCP
Transferencia de archivos	FTP [RFC 959]	TCP
Flujos multimedia	HTTP (por ejemplo, YouTube), RTP	TCP o UDP
Telefonía por Internet	SIP, RTP, o proprietario (por ejemplo, Skype)	Típicamente UDP

Figura 2.5 • Aplicaciones populares de Internet, sus protocolos de la capa de aplicación y sus protocolos de transporte subyacentes.

el Capítulo 7 veremos algunos de estos trucos de diseño. No obstante, un diseño inteligente tiene sus limitaciones cuando el retardo es excesivo, como suele ocurrir a menudo en el caso de la Internet pública. En resumen, actualmente Internet puede ofrecer servicios satisfactorios a las aplicaciones sensibles al tiempo, pero no puede proporcionar ninguna garantía de ancho de banda ni de temporización.

La Figura 2.5 enumera los protocolos de transporte utilizados por algunas aplicaciones populares de Internet. Veremos que aplicaciones como el correo electrónico, el acceso remoto a terminales, la Web y la transferencia de archivos utilizan TCP. Estas aplicaciones han elegido TCP principalmente porque este protocolo ofrece un servicio de transferencia de datos fiable, garantizando que todos los datos llegarán finalmente a su destino. También hemos visto que normalmente la telefonía por Internet se ejecuta sobre UDP. Cada lado de una aplicación de telefonía por Internet necesita enviar datos a través de la red a una cierta velocidad mínima (véase el audio en tiempo real en la Figura 2.4); esto será posible más probablemente con UDP que con TCP. Además, las aplicaciones de telefonía por Internet son tolerantes a las pérdidas de datos, por lo que no necesitan el servicio de transferencia de datos fiable proporcionado por TCP.

Direccionamiento de procesos

Hasta el momento nos hemos centrado en los servicios de transporte existentes entre dos procesos que se comunican. Pero, ¿cómo indica un proceso con qué proceso desea comunicarse utilizando estos servicios? ¿Cómo especifica un proceso que se ejecuta en un host situado en Amherst, Massachusetts, Estados Unidos, que desea comunicarse con un determinado proceso que está ejecutándose en un host ubicado en Bangkok, Tailandia? Para identificar al proceso de recepción, tienen que especificarse dos elementos de información: (1) el nombre o dirección del host y (2) un identificador que especifique el proceso de recepción en el host de destino.

En Internet, el host se identifica mediante su **dirección IP**. En el Capítulo 4 veremos en detalle las direcciones IP. Por el momento, todo lo que necesitamos saber es que una dirección IP es una magnitud de 32 bits que identifica de forma unívoca a un host. (Sin embargo, como veremos en el Capítulo 4, la extendida implantación de los traductores de direcciones

de red (NAT, *Network Address Translators*) ha hecho que, en la práctica, una única dirección IP de 32 bits no defina solamente a un host.)

Además de conocer la dirección del host al que está destinado un mensaje, el host emisor también debe identificar el proceso de recepción que está ejecutándose en el host. Esta información es necesaria porque, en general, un host podría estar ejecutando muchas aplicaciones de red. Un número de puerto de destino sirve a este propósito. Las aplicaciones populares tienen asignados números de puerto específicos. Por ejemplo, un servidor web queda identificado por el número de puerto 80. Un proceso de un servidor de correo (que utilice el protocolo SMTP) se identifica mediante el número de puerto 25. Puede encontrar una lista de números de puerto bien conocidos para todos los protocolos estándar de Internet en http://www.iana.org. Cuando un desarrollador crea una nueva aplicación de red, ésta debe asignarse a un nuevo número de puerto. Examinaremos los números de puerto en detalle en el Capítulo 3.

2.1.5 Protocolos de la capa de aplicación

Acabamos de aprender que los procesos de red se comunican entre sí enviando mensajes a sus sockets. Pero, ¿cómo están estructurados estos mensajes? ¿Cuál es el significado de cada uno de los campos de estos mensajes? ¿Cuándo envían los procesos los mensajes? Estas preguntas nos llevan al ámbito de los protocolos de la capa de aplicación. Un **protocolo de la capa de aplicación** define cómo los procesos de una aplicación, que se ejecutan en distintos sistemas terminales, se pasan los mensajes entre sí. En particular, un protocolo de la capa de aplicación define:

- Los tipos de mensajes intercambiados; por ejemplo, mensajes de solicitud y mensajes de respuesta.
- La sintaxis de los diversos tipos de mensajes, es decir, los campos de los que consta el mensaje y cómo se delimitan esos campos.
- La semántica de los campos, es decir, el significado de la información contenida en los campos.
- Las reglas para determinar cuándo y cómo un proceso envía mensajes y responde a los mismos.

Algunos protocolos de la capa de aplicación están especificados en documentos RFC y, por tanto, son de dominio público. Por ejemplo, el protocolo de la capa de aplicación para la Web, HTTP (*HyperText Transfer Protocol* [RFC 2616]), está disponible como un RFC. Si un navegador web sigue las reglas dadas en el RFC que se ocupa de HTTP, el navegador podrá recuperar páginas web de cualquier servidor web que también se ajuste a dicho RFC. Existen muchos otros protocolos de la capa de aplicación que son propietarios y que intencionadamente no están disponibles para todo el mundo. Por ejemplo, muchos de los sistemas de compartición de archivos P2P existentes utilizan protocolos de la capa de aplicación propietarios.

Es importante diferenciar entre aplicaciones de red y protocolos de la capa de aplicación. Un protocolo de la capa de aplicación es únicamente un elemento de una aplicación de red. Veamos un par de ejemplos. La Web es una aplicación cliente-servidor que permite a los usuarios obtener documentos almacenados en servidores web bajo demanda. La Web consta de muchos componentes, entre los que se incluyen un estándar para los formatos de documentos (es decir, HTML), navegadores web (como Firefox y Microsoft Internet Explo-

rer), servidores web (por ejemplo, servidores Apache y Microsoft) y un protocolo de la capa de aplicación. El protocolo de la capa de aplicación de la Web, HTTP, define el formato y la secuencia de los mensajes que se pasan entre el navegador web y el servidor web. Por tanto, HTTP es sólo una pieza (aunque una pieza importante) de la aplicación web. Otro ejemplo sería una aplicación de correo electrónico Internet, la cual también está constituida por muchos componentes, entre los que se incluyen, los servidores de correo que albergan los buzones de los usuarios; los lectores de correo que permiten a los usuarios leer y crear mensajes; un estándar para definir la estructura de los mensajes de correo electrónico y los protocolos de la capa de aplicación que definen cómo se pasan los mensajes entre los servidores, cómo se pasan los mensajes entre los servidores y los lectores de correo y cómo se interpretan los contenidos de ciertas partes de los mensajes (como por ejemplo, la cabecera). El principal protocolo de la capa de aplicación para el correo electrónico es el protocolo SMTP (*Simple Mail Transfer Protocol*, Protocolo simple de transferencia de correo) [RFC 5321]. Por tanto, el protocolo SMTP sólo es un componente (aunque un componente importante) de la aplicación de correo electrónico.

2.1.6 Aplicaciones de red en este libro

Todos los días se desarrollan nuevas aplicaciones de Internet tanto de dominio público como propietarias. En lugar de abordar un gran número de aplicaciones de Internet a modo de enciclopedia, hemos decidido centrarnos en unas pocas aplicaciones dominantes e importantes. En este capítulo abordaremos cinco aplicaciones importantes: Web, transferencia de archivos, correo electrónico, servicio de directorio y P2P. En primer lugar veremos la Web, no sólo porque es una aplicación enormemente popular, sino porque también su protocolo de la capa de aplicación, HTTP, es sencillo y fácil de comprender. Después de ver esta aplicación, pasaremos a examinar brevemente FTP, porque proporciona un buen contraste con HTTP. A continuación, veremos la aplicación de correo electrónico, que es la aplicación más popular de Internet. El correo electrónico es más complejo que la Web en el sentido de que no utiliza uno sino varios protocolos de la capa de aplicación. A continuación, abordaremos el sistema DNS, que proporciona un servicio de directorio a Internet. La mayoría de los usuarios no interactúan directamente con DNS; en su lugar, invocan indirectamente a DNS a través de otras aplicaciones (entre las que se incluyen las aplicaciones web, de transferencia de archivos y de correo electrónico). DNS ilustra cómo puede implementarse en la capa de aplicación de Internet un elemento de la funcionalidad de red básica (traducción nombre de red a dirección de red). Por último, veremos varias aplicaciones P2P, como la distribución de archivos, las bases de datos distribuidas y la telefonía IP.

2.2 La Web y HTTP

Hasta principios de la década de 1990, Internet era utilizada principalmente por investigadores, profesores y estudiantes universitarios para acceder a hosts remotos, transferir archivos desde los hosts locales a los hosts remotos, y viceversa; y recibir y enviar noticias y mensajes de correo electrónico. Aunque estas aplicaciones eran (y continúan siendo) extremadamente útiles, Internet era prácticamente desconocida fuera de las comunidades académicas y dedicadas a la investigación. Fue entonces, a principios de la década de 1990, cuando una nueva aplicación importante apareció en escena: la World Wide Web [Berners-Lee 1994].

La Web fue la primera aplicación de Internet que atrajo la atención del público general. Cambió de manera dramática, y continúa cambiando, la forma en que las personas interactúan dentro y fuera de sus entornos de trabajo. Hizo que Internet pasará de ser una de las muchas redes de datos a ser prácticamente la única red de datos.

Quizá lo que atrae a la mayoría de los usuarios es que la Web opera *bajo demanda*. Los usuarios reciben lo que desean y cuando lo desean. Es muy diferente a la radio y la televisión, que fuerza a los usuarios a sintonizar los programas cuando el proveedor de contenido tiene disponible el contenido. Además de estar disponible bajo demanda, la Web disfruta de muchas otras maravillosas funciones que a todo el mundo le gustan. Para cualquier individuo es tremendamente fácil publicar información en la Web (todo el mundo puede convertirse en editor con unos costes extremadamente bajos). Los hipervínculos y los motores de búsqueda nos ayudan a navegar a través de un océano de sitios web. Los gráficos estimulan nuestros sentidos. Los formularios, los applets de Java y muchos otros dispositivos nos permiten interactuar con las páginas y sitios. Cada vez más, la Web proporciona una interfaz de menús para grandes cantidades de material de audio y vídeo almacenado en Internet (material multimedia bajo demanda).

2.2.1 Introducción a HTTP

El Protocolo de transferencia de hipertexto (**HTTP**, ***HyperText Transfer Protocol***) es el protocolo de la capa de aplicación de la Web y se encuentra en el corazón de la Web. Está definido en los documentos [RFC 1945] y [RFC 2616]. HTTP se implementa en dos programas: un programa cliente y un programa servidor. El programa cliente y el programa servidor, que se ejecutan en sistemas terminales diferentes, se comunican entre sí intercambiando mensajes HTTP. HTTP define la estructura de estos mensajes y cómo el cliente y el servidor intercambian los mensajes. Antes de explicar en detalle HTTP, vamos a hacer un breve repaso de la terminología Web.

Una **página web** (también denominada documento web) consta de objetos. Un **objeto** es simplemente un archivo (como por ejemplo, un archivo HTML, una imagen JPEG, un applet Java o un clip de vídeo) que puede direccionarse mediante un único URL. La mayoría de las páginas web están constituidas por un **archivo base HTML** y varios objetos referenciados. Por ejemplo, si una página web contiene texto HTML y cinco imágenes JPEG, entonces la página web contiene seis objetos: el archivo base HTML y las cinco imágenes. El archivo base HTML hace referencia a los otros objetos contenidos en la página mediante los URL de los objetos. Cada URL tiene dos componentes: el nombre de host del servidor que alberga al objeto y el nombre de la ruta al objeto. Por ejemplo, en el URL

`http://www.unaEscuela.edu/unDepartmento/imagen.gif`

`www.unaEscuela.edu` corresponde a un nombre de host y `/unDepartmento/imagen.gif` es el nombre de una ruta. Puesto que los **navegadores web** (como Internet Explorer y Firefox) implementan el lado del cliente de HTTP, en el contexto de la Web utilizaremos los términos *navegador* y *cliente* de forma indistinta. Los **servidores web**, que implementan el lado del servidor de HTTP, albergan los objetos web, siendo cada uno de ellos direccionable mediante un URL. Entre los servidores web más populares se incluyen Apache y Microsoft Internet Information Server.

HTTP define cómo los clientes web solicitan páginas web a los servidores web y cómo estos servidores transfieren esas páginas web a los clientes. Más adelante veremos la inte-

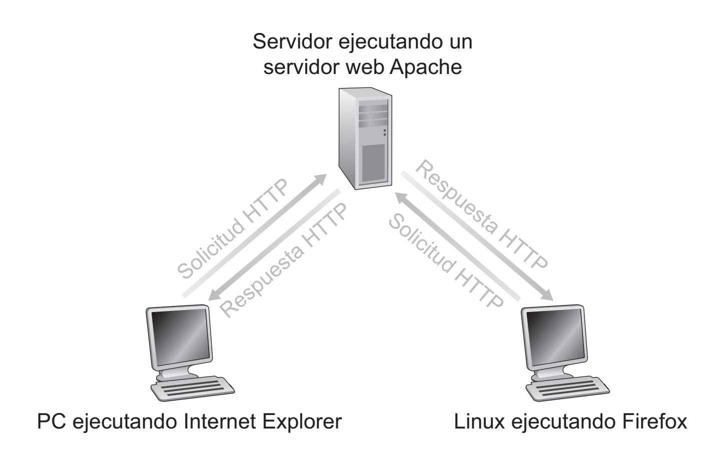


Figura 2.6 • Comportamiento solicitud-respuesta de HTTP.

racción entre el cliente y el servidor en detalle, si bien la idea general se ilustra en la Figura 2.6. Cuando un usuario solicita una página web (por ejemplo, haciendo clic en un hipervínculo), el navegador envía al servidor mensajes de solicitud HTTP para los objetos contenidos en la página. El servidor recibe las solicitudes y responde con mensajes de respuesta HTTP que contienen los objetos.

HTTP utiliza TCP como su protocolo de transporte subyacente (en lugar de ejecutarse por encima de UDP). El cliente HTTP primero inicia una conexión TCP con el servidor. Una vez que la conexión se ha establecido, los procesos de navegador y de servidor acceden a TCP a través de sus interfaces de socket. Como se ha descrito en la Sección 2.1, en el lado del cliente la interfaz del socket es la puerta entre el proceso cliente y la conexión TCP; en el lado del servidor, es la puerta entre el proceso servidor y la conexión TCP. El cliente envía mensajes de solicitud HTTP a su interfaz de socket y recibe mensajes de respuesta HTTP procedentes de su interfaz de socket. De forma similar, el servidor HTTP recibe mensajes de solicitud de su interfaz de socket y envía mensajes de respuesta a través de la interfaz de su socket. Una vez que el cliente envía un mensaje a su interfaz de socket, el mensaje deja de estar "en las manos" del cliente y pasa "a las manos" de TCP. Como hemos visto en la Sección 2.1, TCP proporciona un servicio de transferencia de datos fiable a HTTP. Esto implica que cada mensaje de solicitud HTTP enviado por un proceso cliente llegará intacto al servidor; del mismo modo, cada mensaje de respuesta HTTP enviado por el proceso servidor llegará intacto al cliente. Esta es una de las grandes ventajas de una arquitectura en capas: HTTP no tiene que preocuparse por las pérdidas de datos o por los detalles sobre cómo TCP recupera los datos perdidos o los reordena dentro de la red. Éste es el trabajo de TCP y de los protocolos de las capas inferiores de la pila de protocolos.

Es importante observar que el servidor envía los archivos solicitados a los clientes sin almacenar ninguna información acerca del estado del cliente. Si un determinado cliente pide el mismo objeto dos veces en un espacio de tiempo de unos pocos segundos, el servidor no responde diciendo que acaba de servir dicho objeto al cliente; en su lugar, el servidor reenvía el objeto, ya que ha olvidado por completo que ya lo había hecho anteriormente. Dado que un servidor HTTP no mantiene ninguna información acerca de los clientes, se dice que HTTP es un **protocolo sin memoria del estado**. Debemos destacar también que la Web utiliza la arqui-

tectura de aplicación cliente-servidor, como se ha explicado en la Sección 2.1. Un servidor web siempre está activo y tiene una dirección IP fija y da servicio a solicitudes procedentes de, potencialmente, millones de navegadores distintos.

2.2.2 Conexiones persistentes y no persistentes

En muchas aplicaciones de Internet, el cliente y el servidor están en comunicación durante un periodo de tiempo amplio, haciendo el cliente una serie de solicitudes y el servidor respondiendo a dichas solicitudes. Dependiendo de la aplicación y de cómo se esté empleando la aplicación, las solicitudes pueden hacerse una tras otra, periódicamente a intervalos regulares o de forma intermitente. Cuando esta interacción cliente-servidor tiene lugar sobre TCP, el desarrollador de la aplicación tiene que tomar una decisión importante: ¿debería cada par solicitud/respuesta enviarse a través de una conexión TCP *separada* o deberían enviarse todas las solicitudes y sus correspondientes respuestas a través de la *misma* conexión TCP? Si se utiliza el primer método, se dice que la aplicación utiliza **conexiones no persistentes**; si se emplea la segunda opción, entonces se habla de **conexiones persistentes**. Con el fin de profundizar en esta cuestión de diseño, vamos a examinar las ventajas y desventajas de las conexiones persistentes en el contexto de una aplicación específica, como por ejemplo HTTP, que puede utilizar ambos tipos de conexión. Aunque HTTP emplea conexiones persistentes en su modo por defecto, los clientes y servidores HTTP se pueden configurar para emplear en su lugar conexiones no persistentes.

HTTP con conexiones no persistentes

Sigamos los pasos que permiten transferir una página web desde un servidor a un cliente en el caso de conexiones no persistentes. Supongamos que la página consta de un archivo base HTML y de 10 imágenes JPEG, residiendo los 11 objetos en el mismo servidor. Supongamos también que el URL del archivo base HTML es:

```
http://www.unaEscuela.edu/unDepartmento/home.index
```

Lo que ocurre es lo siguiente:

1. El proceso cliente HTTP inicia una conexión TCP con el servidor www.unaEscuela.edu en el puerto número 80, que es el número de puerto por defecto para HTTP. Asociados con la conexión TCP, habrá un socket en el cliente y un socket en el servidor.
2. El cliente HTTP envía un mensaje de solicitud HTTP al servidor a través de su socket. El mensaje de solicitud incluye el nombre de la ruta `/unDepartmento/home.index`. (Más adelante veremos en detalle los mensajes HTTP.)
3. El proceso servidor HTTP recibe el mensaje de solicitud a través de su socket, recupera el objeto `/unDepartmento/home.index` de su medio de almacenamiento (RAM o disco), encapsula el objeto en un mensaje de respuesta HTTP y lo envía al cliente a través de su socket.
4. El proceso servidor HTTP indica a TCP que cierre la conexión TCP. (Pero TCP realmente no termina la conexión hasta que está seguro de que el cliente ha recibido el mensaje de respuesta en perfecto estado.)
5. El cliente HTTP recibe el mensaje de respuesta. La conexión TCP termina. El mensaje indica que el objeto encapsulado es un archivo HTML. El cliente extrae el archivo del

mensaje de respuesta, examina el archivo HTML y localiza las referencias a los 10 objetos JPEG.

6. Los cuatro primeros pasos se repiten entonces para cada uno de los objetos JPEG referenciados.

Cuando el navegador recibe la página web, la muestra al usuario. Dos navegadores distintos pueden interpretar (es decir, mostrar al usuario) una página web de formas distintas. HTTP no tiene nada que ver con cómo un cliente interpreta una página web. Las especificaciones HTTP ([RFC 1945] y [RFC 2616]) únicamente definen el protocolo de comunicación entre el programa HTTP cliente y el programa HTTP servidor.

Los pasos anteriores ilustran el uso de las conexiones no persistentes, donde cada conexión TCP se cierra después de que el servidor envíe el objeto (la conexión no se mantiene para los restantes objetos). Observe que cada conexión TCP transporta exactamente un mensaje de solicitud y un mensaje de respuesta. Por tanto, en este ejemplo, cuando un usuario solicita la página web, se generan 11 conexiones TCP.

En los pasos descritos anteriormente, hemos sido intencionadamente vagos en lo que respecta a si el cliente obtiene las diez imágenes JPEG a través de diez conexiones serie TCP o si algunas de dichas imágenes se obtienen a través de conexiones TCP paralelo. De hecho, los usuarios pueden configurar los navegadores modernos para controlar el grado de paralelismo. En sus modos por defecto, la mayoría de los navegadores abren de 5 a 10 conexiones TCP paralelo y cada una de estas conexiones gestiona una transacción solicitud-respuesta. Si el usuario lo prefiere, el número máximo de conexiones en paralelo puede establecerse en uno, en cuyo caso se establecerán diez conexiones en serie. Como veremos en el siguiente capítulo, el uso de conexiones paralelo reduce el tiempo de respuesta.

Antes de continuar, vamos a realizar un cálculo aproximado para estimar la cantidad de tiempo que transcurre desde que un cliente solicita el archivo base HTML hasta que recibe dicho archivo completo. Para ello, definimos el **tiempo de ida y vuelta** (**RTT, *Round-Trip Time***), que es el tiempo que tarda un paquete pequeño en viajar desde el cliente al servidor y volver de nuevo al cliente. El tiempo RTT incluye los retardos de propagación de los paquetes, los retardos de cola en los routers y switches intermedios y los retardos de procesamiento de los paquetes (estos retardos se explican en la Sección 1.4). Consideremos ahora lo que ocurre cuando un usuario hace clic en un hipervínculo. Como se muestra en la Figura 2.7, esto hace que el navegador inicie una conexión TCP entre el navegador y el servidor web, lo que implica un proceso de "acuerdo en tres fases" (el cliente envía un pequeño segmento TCP al servidor, el servidor reconoce y responde con otro pequeño segmento TCP y, por último, el cliente devuelve un mensaje de reconocimiento al servidor). Las dos primeras partes de este proceso de acuerdo en tres fases tardan un periodo de tiempo RTT. Después de completarse las dos primeras fases de la negociación, el cliente envía a la conexión TCP el mensaje de solicitud HTTP combinado con la tercera parte de la negociación (el mensaje de reconocimiento). Una vez que el mensaje de solicitud llega al servidor, éste envía el archivo HTML a la conexión TCP. Este mensaje de solicitud/respuesta HTTP consume otro periodo de tiempo RTT. Luego el tiempo de respuesta total es aproximadamente igual a dos RTT más el tiempo de transmisión del archivo HTML en el servidor.

HTTP con conexiones persistentes

Las conexiones no persistentes presentan algunos inconvenientes. En primer lugar, tiene que establecerse y mantenerse una conexión completamente nueva para *cada objeto soli-*

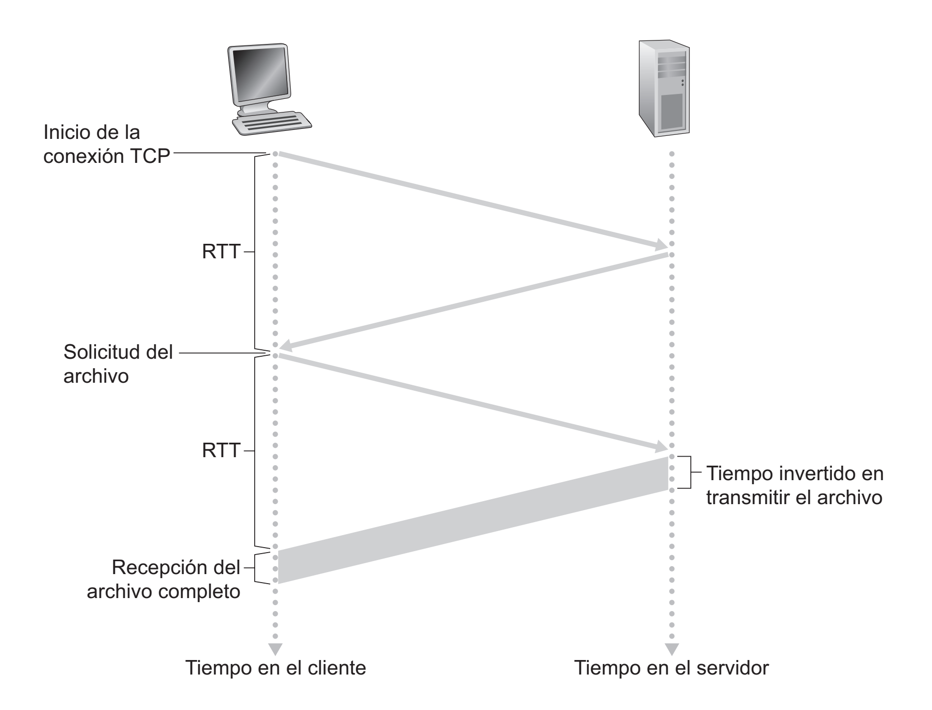


Figura 2.7 • Cálculo aproximado del tiempo necesario para solicitar y recibir un archivo HTML.

citado. Para cada una de estas conexiones, deben asignarse los buffers TCP y las variables TCP tienen que mantenerse tanto en el cliente como en el servidor. Esto puede sobrecargar de forma significativa al servidor web, ya que puede estar sirviendo solicitudes de cientos de clientes distintos simultáneamente. En segundo lugar, como ya hemos explicado, cada objeto sufre un retardo de entrega de dos RTT: un RTT para establecer la conexión TCP y otro RTT para solicitar y recibir un objeto.

Con las conexiones persistentes, el servidor deja la conexión TCP abierta después de enviar una respuesta. Las subsiguientes solicitudes y respuestas que tienen lugar entre el mismo cliente y el servidor pueden enviarse a través de la misma conexión. En concreto, una página web completa (en el ejemplo anterior, el archivo base HTML y las 10 imágenes) se puede enviar a través de una misma conexión TCP persistente. Además, varias páginas web que residan en el mismo servidor pueden enviarse desde el servidor a un mismo cliente a través de una única conexión TCP persistente. Estas solicitudes de objetos pueden realizarse una tras otra sin esperar a obtener las respuestas a las solicitudes pendientes (*pipelining*, procesamiento en cadena). Normalmente, el servidor HTTP cierra una conexión cuando no se ha utilizado durante cierto tiempo (un intervalo de fin de temporización configurable). Cuando el servidor recibe solicitudes una tras otra, envía los objetos uno tras otro. El modo por defecto de HTTP utiliza conexiones persistentes con procesamiento en cadena. En los problemas de repaso de los Capítulos 2 y 3 compararemos cuantitativamente el rendimiento de las conexiones persistentes y no persistentes. Le animamos también a que consulte [Heidemann 1997; Nielsen 1997].

2.2.3 Formato de los mensajes HTTP

Las especificaciones HTTP [RFC 2616]) incluyen las definiciones de los formatos de los mensajes HTTP. A continuación vamos a estudiar los dos tipos de mensajes HTTP existentes: mensajes de solicitud y mensajes de respuesta.

Mensaje de solicitud HTTP

A continuación le proporcionamos un mensaje de solicitud HTTP típico:

```
GET /unadireccion/pagina.html HTTP/1.1
Host: www.unaescuela.edu
Connection: close
User-agent: Mozilla/4.0
Accept-language: fr
```

Podemos aprender muchas cosas si miramos en detalle este sencillo mensaje de solicitud. En primer lugar, podemos comprobar que el mensaje está escrito en texto ASCII normal, por lo que cualquier persona con conocimientos informáticos puede leerlo. En segundo lugar, vemos que el mensaje consta de cinco líneas, cada una de ellas seguida por un retorno de carro y un salto de línea. La última línea va seguida de un retorno de carro y un salto de línea adicionales. Aunque este mensaje en concreto está formado por cinco líneas, un mensaje de solicitud puede constar de muchas más líneas o tener tan pocas como únicamente una. La primera línea de un mensaje de solicitud HTTP se denomina **línea de solicitud** y las siguientes líneas son las **líneas de cabecera**. La línea de solicitud consta de tres campos: el campo de método, el campo URL y el campo de la versión HTTP. El campo que especifica el método puede tomar diferentes valores, entre los que se incluyen `GET, POST, HEAD, PUT` y `DELETE.` La inmensa mayoría de los mensajes de solicitud HTTP utilizan el método `GET`. Este método se emplea cuando el navegador solicita un objeto, identificando dicho objeto en el campo URL. En este ejemplo, el navegador está solicitando el objeto `/unadireccion/pagina. html.` El campo correspondiente a la versión se explica por sí mismo; en este ejemplo, el navegador utiliza la versión HTTP/1.1.

Analicemos ahora las líneas de cabecera de este ejemplo. La línea de cabecera `Host: www.unaescuela.edu` especifica el host en el que reside el objeto. Podría pensarse que esta línea de cabecera es innecesaria, puesto que ya existe una conexión TCP activa con el host. Pero, como veremos en la Sección 2.2.5, las cachés proxy web necesitan la información proporcionada por la línea de cabecera del host. Al incluir la línea de cabecera `Connection: close`, el navegador está diciendo al servidor que no desea molestarse en trabajar con conexiones persistentes, sino que desea que el servidor cierre la conexión después de enviar el objeto solicitado. La línea de cabecera `User-agent:` especifica el agente de usuario, es decir, el tipo de navegador que está haciendo la solicitud al servidor. En este caso, el agente de usuario es Mozilla/4.0, un navegador de Netscape. Esta línea de cabecera resulta útil porque el servidor puede enviar versiones diferentes del mismo objeto a los distintos tipos de agentes de usuario (el mismo URL direcciona a cada una de las versiones). Por último, la línea de cabecera `Accept-language:` indica que el usuario prefiere recibir una versión en francés del objeto, si tal objeto existe en el servidor; en caso contrario, el servidor enviará la versión por defecto. La línea de cabecera `Accept-language:` sólo es una de las muchas cabeceras de negociación del contenido disponibles en HTTP.

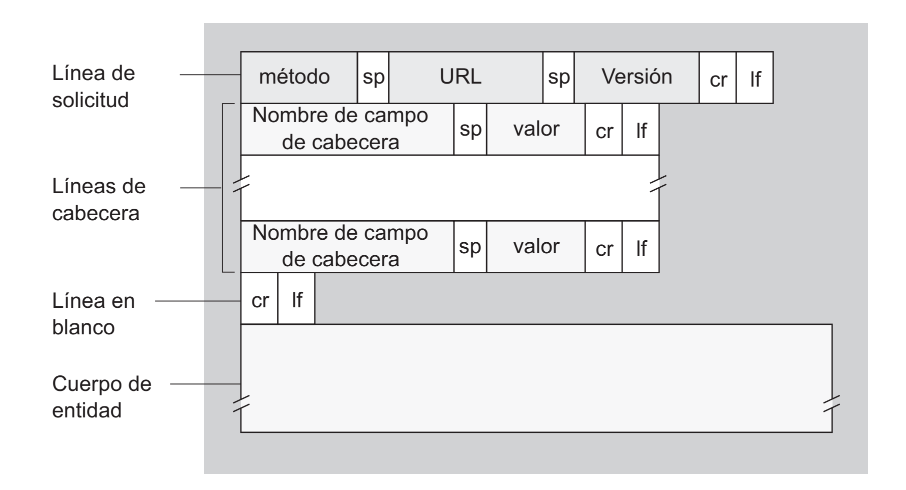


Figura 2.8 • Formato general de un mensaje de solicitud HTTP.

Una vez visto este ejemplo, vamos a estudiar el formato general de un mensaje de solicitud como el de la Figura 2.8. Podemos comprobar que el formato general es muy similar al usado en el ejemplo anterior. Sin embargo, fíjese en que después de las líneas de cabecera (y el retorno de carro y el salto de línea adicionales) se incluye un "cuerpo de entidad". Este campo queda vacío cuando se utiliza el método `GET`, pero no cuando se usa el método `POST`. A menudo, un cliente HTTP utiliza el método `POST` cuando el usuario completa un formulario; por ejemplo, cuando especifica términos para realizar una búsqueda utilizando un motor de búsqueda. Con un mensaje `POST`, el usuario solicita una página web al servidor, pero el contenido concreto de la misma dependerá de lo que el usuario haya escrito en los campos del formulario. Si el valor del campo método es `POST`, entonces el cuerpo de la entidad contendrá lo que el usuario haya introducido en los campos del formulario.

Seríamos descuidados si no dijéramos que una solicitud generada con un formulario no necesariamente utiliza el método `POST`. En su lugar, a menudo los formularios HTML emplean el método `GET` e incluyen los datos de entrada (especificados en los campos del formulario) en el URL solicitado. Por ejemplo, si un formulario emplea el método `GET` y tiene dos campos, y las entradas a esos dos campos son `monkeys` y `bananas`, entonces el URL tendrá la estructura `www.unsitio.com/animalsearch?monkeys&bananas`. Probablemente, al navegar por la Web habrá visto direcciones URL de este tipo.

El método `HEAD` es similar al método `GET`. Cuando un servidor recibe una solicitud con el método `HEAD`, responde con un mensaje HTTP, pero excluye el objeto solicitado. Los desarrolladores de aplicaciones a menudo utilizan el método `HEAD` para labores de depuración. El método `PUT` suele utilizarse junto con herramientas de publicación web. Esto permite a un usuario cargar un objeto en una ruta específica (directorio) en un servidor web determinado. Las aplicaciones que necesitan cargar objetos en servidores web también emplean el método `PUT`. El método `DELETE` permite a un usuario o a una aplicación borrar un objeto de un servidor web.

Mensajes de respuesta HTTP

A continuación proporcionamos un mensaje de respuesta HTTP típico. Este mensaje de respuesta podría ser la respuesta al mensaje de solicitud ejemplo que acabamos de ver.

```
HTTP/1.1 200 OK
Connection: close
Date: Sat, 07 Jul 2007 12:00:15 GMT
Server: Apache/1.3.0 (Unix)
Last-Modified: Sun, 6 May 2007 09:23:24 GMT
Content-Length: 6821
Content-Type: text/html

(datos datos datos datos datos ...)
```

Examinemos detenidamente este mensaje de respuesta. Tiene tres secciones: una **línea de estado** inicial, seis **líneas de cabecera** y el **cuerpo de entidad**. El cuerpo de entidad es la parte más importante del mensaje, ya que contiene el objeto solicitado en sí (representado por la línea `datos datos datos datos datos ...`). La línea de estado contiene tres campos: el que especifica la versión del protocolo, el correspondiente al código de estado y el tercero que contiene el mensaje explicativo del estado correspondiente. En este ejemplo, la línea de estado indica que el servidor está utilizando HTTP/1.1 y que todo es correcto (OK); es decir, que el servidor ha encontrado y está enviando el objeto solicitado.

Veamos ahora las líneas de cabecera. El servidor utiliza la línea `Connection: close` para indicar al cliente que va a cerrar la conexión TCP después de enviar el mensaje. La línea de cabecera `Date:` indica la hora y la fecha en la que se creó la respuesta HTTP y fue enviada por el servidor. Observe que no especifica la hora en que el objeto fue creado o modificado por última vez; es la hora en la que el servidor recupera el objeto de su sistema de archivos, inserta el objeto en el mensaje de respuesta y lo envía. La línea de cabecera `Server:` indica que el mensaje fue generado por un servidor web Apache; ésta es análoga a la línea de cabecera `User-agent:` del mensaje de solicitud HTTP. La línea `Last-Modified:` especifica la hora y la fecha en que el objeto fue creado o modificado por última vez. La línea de cabecera `Last-Modified:`, que enseguida estudiaremos en detalle, resulta fundamental para el almacenamiento en caché del objeto, tanto en el cliente local como en los servidores de almacenamiento en caché de la red (también conocidos como servidores proxy). La línea `Content-Length:` especifica el número de bytes del objeto que está siendo enviado. La línea `Content-Type:` indica que el objeto especificado en el cuerpo de entidad es texto HTML. (El tipo de objeto está indicado oficialmente por la línea de cabecera `Content-Type:` y no por la extensión del archivo.).

Una vez visto un ejemplo, vamos a pasar a examinar el formato general de un mensaje de respuesta, como el mostrado en la Figura 2.9. Este formato general de mensaje de respuesta se corresponde con el del ejemplo anterior. Comentaremos algunas cosas acerca de los códigos de estado y sus descripciones. El código de estado y su frase asociada indican el resultado de la solicitud. Algunos de los códigos de estado y sus frases asociadas son:

- `200 OK:` La solicitud se ha ejecutado con éxito y se ha devuelto la información en el mensaje de respuesta.
- `301 Moved Permanently:` El objeto solicitado ha sido movido de forma permanente; el nuevo URL se especifica en la línea de cabecera `Location:` del mensaje de respuesta. El software cliente recuperará automáticamente el nuevo URL.
- `400 Bad Request:` Se trata de un código de error genérico que indica que la solicitud no ha sido comprendida por el servidor.

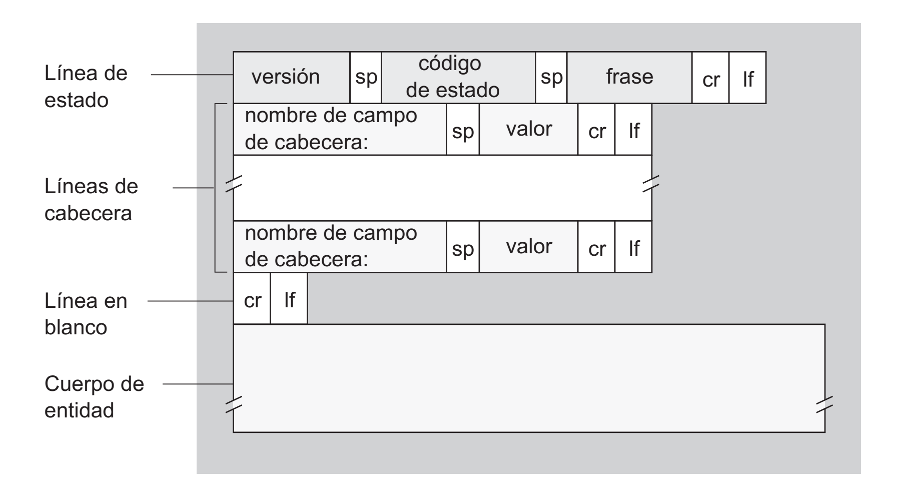


Figura 2.9 • Formato general de un mensaje de respuesta HTTP.

- `404 Not Found`: El documento solicitado no existe en este servidor.
- `505 HTTP Version Not Supported`: La versión de protocolo HTTP solicitada no es soportada por el servidor.

¿Le gustaría ver un mensaje de respuesta HTTP real? Esto es muy recomendable y además es muy fácil de hacer. En primer lugar, establezca una conexión Telnet con su servidor web favorito. A continuación escriba un mensaje de solicitud de una línea para obtener algún objeto que esté almacenado en ese servidor. Por ejemplo, si tiene acceso a la línea de comandos (*prompt*), escriba:

```
telnet cis.poly.edu 80

GET /~ross/ HTTP/1.1
Host: cis.poly.edu
```

(Pulse dos veces la tecla retorno de carro después de escribir la última línea.) De este modo se abre una conexión TCP en el puerto 80 del host `cis.poly.edu` y luego se envía el mensaje de solicitud HTTP. Debería ver un mensaje de respuesta que incluya el archivo base HTML de la página inicial del profesor Ross. Para ver simplemente las líneas del mensaje HTTP y no recibir el objeto, sustituya `GET` por `HEAD`. Por último, sustituya `/~ross/` por `/~banana/` y fíjese en el tipo de mensaje de respuesta que recibe.

En esta sección hemos visto una serie de líneas de cabecera que pueden utilizarse en los mensajes HTTP de solicitud y respuesta. La especificación de HTTP define un gran número de otras líneas de cabecera que pueden ser insertadas por los navegadores, servidores web y servidores de almacenamiento en caché de la red. Hemos cubierto únicamente una pequeña parte de la totalidad disponible de líneas de cabecera. A continuación veremos unas pocas más y en la Sección 2.2.5 algunas otras, al hablar del almacenamiento en cachés web de la red. Puede leer una exposición enormemente clara y comprensible acerca del protocolo HTTP, incluyendo sus cabeceras y códigos de estado en [Krishnamurty 2001]; consulte también [Luotonen 1998] para ver una exposición desde el punto de vista del desarrollador.

¿Cómo decide un navegador qué líneas de cabecera incluir en un mensaje de solicitud? ¿Cómo decide un servidor web qué líneas de cabecera incluir en un mensaje de respuesta? Un navegador generará líneas de cabecera en función del tipo y la versión del navegador (por ejemplo, un navegador HTTP/1.0 no generará ninguna línea de cabecera correspondiente a la versión 1.1), de la configuración del navegador que tenga el usuario (por ejemplo, el idioma preferido) y de si el navegador actualmente tiene en caché una versión del objeto, posiblemente desactualizada. Los servidores web se comportan de forma similar: existen productos, versiones y configuraciones diferentes, que influyen en las líneas de cabecera que se incluirán en los mensajes de respuesta.

TP 5.1

2.2.4 Interacción usuario-servidor: cookies

Hemos mencionado anteriormente que un servidor HTTP no tiene memoria del estado de la conexión. Esto simplifica el diseño del servidor y ha permitido a los ingenieros desarrollar servidores web de alto rendimiento que pueden gestionar miles de conexiones TCP simultáneas. Sin embargo, para un sitio web, a menudo es deseable poder identificar a los usuarios, bien porque el servidor desea restringir el acceso a los usuarios o porque desea servir el contenido en función de la identidad del usuario. Para estos propósitos, HTTP utiliza cookies. Las cookies, definidas en el documento RFC 2965, permiten a los sitios seguir la pista a los usuarios. Actualmente, la mayoría de los sitios web comerciales más importantes utilizan cookies.

Como se muestra en la Figura 2.10, la tecnología de las cookies utiliza cuatro componentes: (1) una línea de cabecera de la cookie en el mensaje de respuesta HTTP; (2) una línea de cabecera de la cookie del mensaje de solicitud HTTP; (3) el archivo de cookies almacenado en el sistema terminal del usuario y gestionado por el navegador del usuario; (4) una base de datos back-end en el sitio web. Basándonos en la Figura 2.10, vamos a ver mediante un ejemplo cómo funcionan las cookies. Suponga que Susana, que accede siempre a la Web utilizando Internet Explorer en el PC de su casa, entra en Amazon.com por primera vez. Supongamos que anteriormente ella había visitado el sitio de eBay. Cuando la solicitud llega al servidor web de Amazon, el servidor crea un número de identificación único y crea una entrada en su base de datos back-end que está indexada por el número de identificación. El servidor web de Amazon responde entonces al navegador de Susana, incluyendo en la respuesta HTTP una línea de cabecera `Set-cookie:`, que contiene el número de identificación. Por ejemplo, la línea de cabecera podría ser:

```
Set-cookie: 1678
```

Cuando el navegador de Susana recibe el mensaje de respuesta HTTP, ve la cabecera `Set-cookie:`. Entonces el navegador añade una línea a su archivo especial de cookies. Esta línea incluye el nombre de host del servidor y el número de identificación en la cabecera `Set-cookie:`. Observe que el archivo de cookies ya tiene una entrada para eBay, dado que Susana había visitado dicho sitio en el pasado. A medida que Susana continúa navegando por el sitio de Amazon, cada vez que solicita una página web su navegador consulta su archivo de cookies, extrae su número de identificación para ese sitio y añade una cabecera de cookie que incluye el número en identificación de la solicitud HTTP. Específicamente, cada una de sus solicitudes HTTP al servidor de Amazon incluyen la línea de cabecera:

```
Cookie: 1678
```

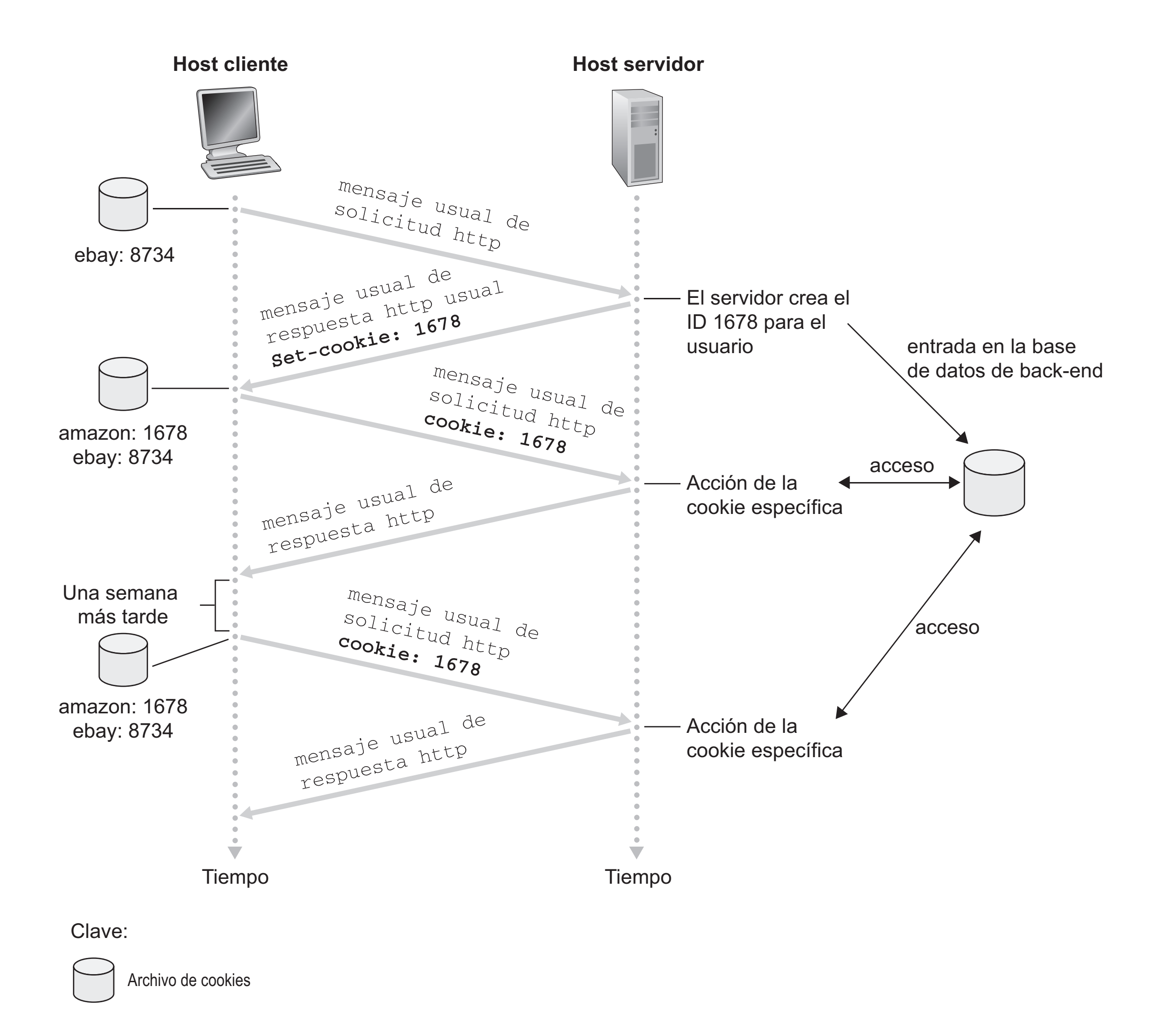


Figura 2.10 • Mantenimiento del estado del usuario mediante cookies.

De esta forma, el servidor de Amazon puede seguir la actividad de Susana en su sitio. Aunque el sitio web de Amazon no necesariamente conoce el nombre de Susana, sabe exactamente qué páginas ha visitado el usuario número 1678, en qué orden y cuántas veces. Amazon utiliza cookies para proporcionar su servicio de carro de la compra; Amazon puede mantener una lista de todas las compras de Susana, con el fin de que ella pueda pagarlas todas juntas al final de la sesión.

Si Susana vuelve al sitio de Amazon, por ejemplo, una semana más tarde, su navegador continuará incluyendo la línea de cabecera `Cookie: 1678` en los mensajes de solicitud. Amazon también recomienda productos a Susana basándose en las páginas web que ha visitado anteriormente dentro del sitio. Si Susana se registra en Amazon, proporcionando su nombre completo, dirección de correo electrónico, dirección postal y la información de su tarjeta crédito, entonces Amazon podrá incluir esta información en su base de datos, asociando el nombre de Susana con su número de identificación (¡y todas las páginas que ha

visitado dentro del sitio en el pasado!). Así es como Amazon y otros sitios de comercio electrónico proporcionan el servicio de "compra con un clic"; cuando Susana desee comprar un producto en una visita posterior, no tendrá que volver a escribir su nombre, el número de la tarjeta de crédito ni su dirección.

Por tanto, podemos afirmar que las cookies pueden utilizarse para identificar a un usuario. La primera vez que un usuario visita un sitio, el usuario puede proporcionar una identificación del mismo (posiblemente su nombre). Así, en las sesiones posteriores, el navegador pasa una cabecera de cookie al servidor, identificando al usuario ante el servidor. Las cookies pueden por tanto utilizarse para crear una capa de sesión por encima del protocolo HTTP sin memoria del estado de la conexión. Por ejemplo, cuando un usuario inicia una sesión en una aplicación de correo electrónico basada en la Web (como por ejemplo Hotmail), el navegador envía al servidor la información de la cookie, permitiendo al servidor identificar al usuario a lo largo de la sesión que el usuario mantenga con la aplicación.

Aunque las cookies a menudo simplifican a los usuarios la realización de compras por Internet, son controvertidas, porque también pueden considerarse como una invasión de la intimidad del usuario. Como acabamos de ver, empleando una combinación de cookies y de la información sobre cuentas suministrada por el usuario, un sitio web puede obtener mucha información de un usuario y, potencialmente, puede vender esa información a otras empresas. Cookie Central [Cookie Central 2008] proporciona mucha información acerca de la controversia de las cookies.

2.2.5 Almacenamiento en caché web

Una **caché web**, también denominada **servidor proxy**, es una entidad de red que satisface solicitudes HTTP en nombre de un servidor web de origen. La caché web dispone de su propio almacenamiento en disco y mantiene en él copias de los objetos solicitados recientemente. Como se muestra en la Figura 2.11, el navegador de un usuario se puede configurar de modo que todas sus solicitudes HTTP se dirijan en primer lugar a la caché web. Por ejemplo, suponga que un navegador está solicitando el objeto `http://www.unaEscuela.edu/campus.gif`. Veamos qué es lo que ocurre:

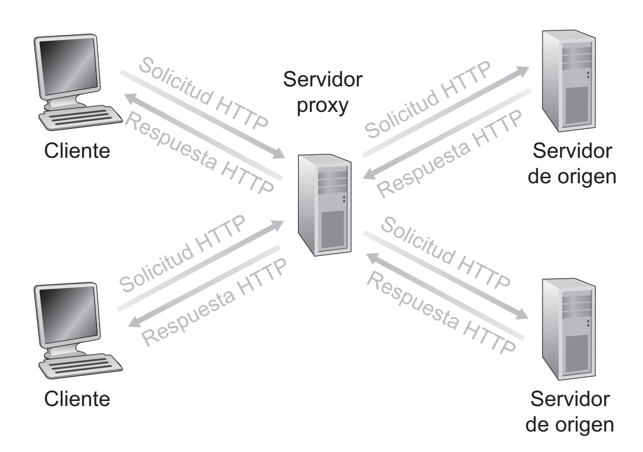


Figura 2.11 • Clientes que solicitan objetos a través de una caché web.

1. El navegador establece una conexión TCP con la caché web y envía una solicitud HTTP para el objeto a la caché web.
2. La caché web comprueba si tiene una copia del objeto almacenada localmente. Si la tiene, la caché web devuelve el objeto dentro de un mensaje de respuesta HTTP al navegador del cliente.
3. Si la caché web no tiene el objeto, abre una conexión TCP con el servidor de origen, es decir, con www.unaEscuela.edu. La caché web envía entonces una solicitud HTTP para obtener el objeto a través de la conexión TCP caché-servidor. Después de recibir esta solicitud, el servidor de origen envía el objeto dentro de un mensaje de respuesta HTTP a la caché web.
4. Cuando la caché web recibe el objeto, almacena una copia en su dispositivo de almacenamiento local y envía una copia, dentro de un mensaje de respuesta HTTP, al navegador del cliente (a través de la conexión TCP existente entre el navegador del cliente y la caché web).

Observe que una caché es a la vez un servidor y un cliente. Cuando recibe solicitudes de y envía respuestas a un navegador, se comporta como un servidor. Cuando envía solicitudes a y recibe respuestas de un servidor de origen, entonces actúa como un cliente.

Habitualmente es un ISP quien adquiere e instala una caché web. Por ejemplo, una universidad podría instalar una caché en su red del campus y configurar todos los navegadores del campus apuntando a la caché. O un ISP residencial de gran tamaño (como AOL) podría instalar una o más cachés en su red y preconfigurar los navegadores que suministre para que apunten a las cachés instaladas.

El almacenamiento en caché web se ha implantado en Internet por dos razones. La primera razón es que una caché web puede reducir sustancialmente el tiempo de respuesta a la solicitud de un cliente, especialmente si el ancho de banda cuello de botella entre el cliente y el servidor de origen es mucho menor que el ancho de banda cuello de botella entre el cliente y la caché. Si existe una conexión de alta velocidad entre el cliente y la caché, lo que ocurre a menudo, y si la caché tiene el objeto solicitado, entonces ésta podrá suministrar el objeto rápidamente al cliente. La segunda razón es que, como ilustraremos enseguida con un ejemplo, las cachés web pueden reducir sustancialmente el tráfico en el enlace de acceso a Internet de una institución. Reduciendo el tráfico, la institución (por ejemplo, una empresa o una universidad) no tiene que mejorar el ancho de banda tan rápidamente, lo que implica una reducción de costes. Además, las cachés web pueden reducir mucho el tráfico web global en Internet, mejorando en consecuencia el rendimiento de todas las aplicaciones.

Para adquirir un conocimiento profundo de las ventajas de las cachés, vamos a ver un ejemplo en el contexto ilustrado en la Figura 2.12. Esta figura muestra dos redes: la red institucional y el resto de la red pública Internet. La red institucional es una LAN de alta velocidad. Un router de la red institucional y un router de Internet están conectados mediante un enlace a 15 Mbps. Los servidores de origen están conectados a Internet pero se encuentran distribuidos por todo el mundo. Suponga que el tamaño medio del objeto es de 1 Mbits y que la tasa media de solicitudes de los navegadores de la institución a los servidores de origen es de 15 solicitudes por segundo. Suponga que los mensajes de solicitud HTTP son extremadamente pequeños y que por tanto no crean tráfico en las redes ni en el enlace de acceso (desde el router institucional al router de Internet). Suponga también que el tiempo que se tarda en reenviar una solicitud HTTP (contenida en un datagrama IP) desde el router del lado de Internet del enlace de acceso de la Figura 2.12 hasta que se recibe la respuesta

(normalmente contenida en muchos datagramas IP) es igual a dos segundos. Informalmente, vamos a denominar a este último retardo "retardo de Internet".

El tiempo total de respuesta, es decir, el tiempo desde que el navegador solicita un objeto hasta que lo recibe, es la suma del retardo de la LAN, el retardo de acceso (es decir, el retardo entre los dos routers) y el retardo de Internet. Hagamos ahora un burdo cálculo para estimar este retardo. La intensidad de tráfico en la LAN es (véase la Sección 1.4.2):

$$(15 \text{ solicitudes/seg}) \cdot (1 \text{ Mbits/solicitud})/(100 \text{ Mbps}) = 0{,}15$$

mientras que la intensidad de tráfico en el enlace de acceso (desde el router de Internet hasta el router de la institución) es:

$$(15 \text{ solicitudes/seg}) \cdot (1 \text{ Mbits/solicitud})/(15 \text{ Mbps}) = 1$$

Una intensidad de tráfico de 0,15 en una LAN normalmente da resultados, como máximo, de decenas de milisegundos de retardo; por tanto, podemos despreciar el retardo de la LAN. Sin embargo, como hemos visto en la Sección 1.4.2, cuando la intensidad de tráfico se aproxima a 1 (como es el caso del enlace de acceso de la Figura 2.12), el retardo en el enlace comienza a aumentar y crece sin límite. Por tanto, el tiempo medio de respuesta para satisfacer las solicitudes es del orden de minutos, si no mayor, lo que es inaceptable para los usuarios de la institución. Evidentemente, es necesario hacer algo.

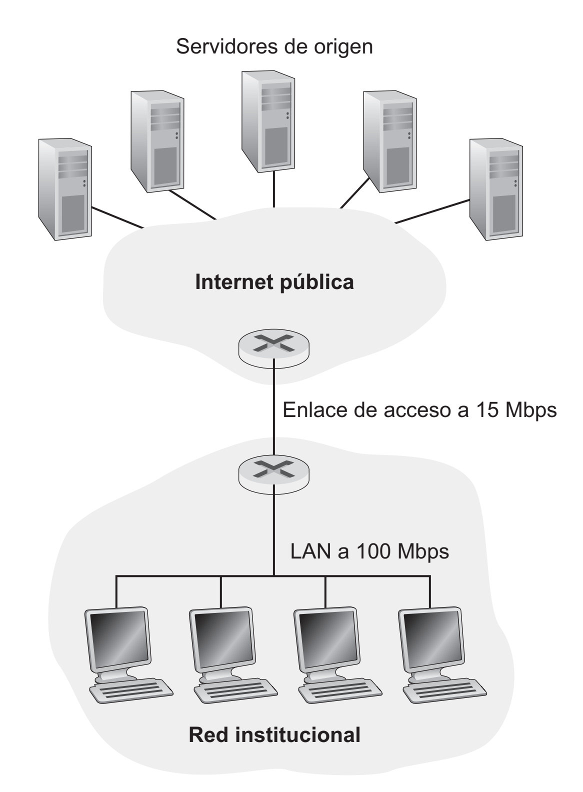


Figura 2.12 • Cuello de botella entre una red institucional e Internet.

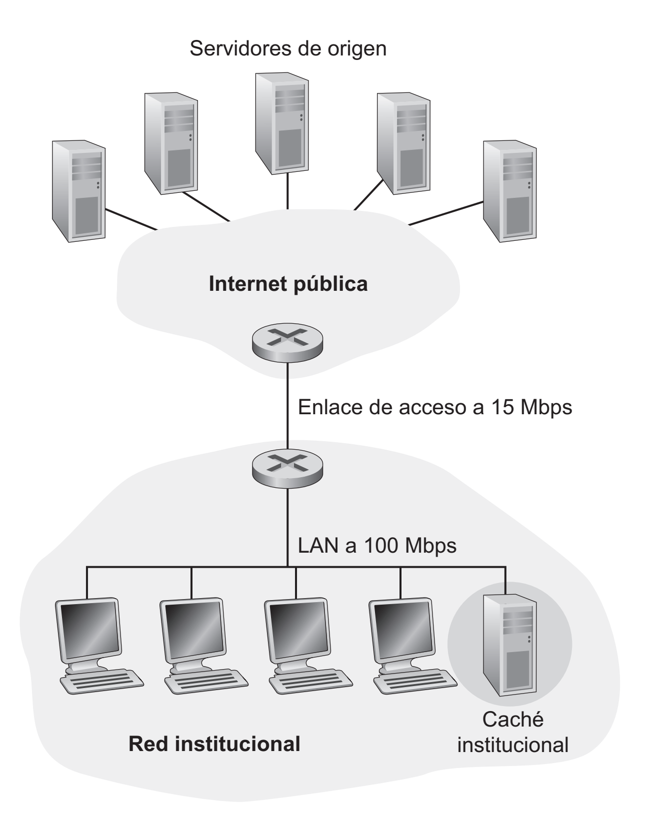


Figura 2.13 • Adición de una caché a una red institucional.

Una posible solución es incrementar la velocidad de acceso de 15 Mbps a, por ejemplo, 100 Mbps. Esto disminuirá la intensidad de tráfico en el enlace de acceso a 0,15, lo que se traduce en retardos despreciables entre los dos routers. En este caso, el tiempo total de respuesta será de unos dos segundos, es decir, el retardo de Internet. Pero esta solución también implica que la institución debe actualizar su enlace de acceso de 15 Mbps a 100 Mbps, lo que resulta ser una solución cara.

Consideremos ahora la solución alternativa de no actualizar el enlace de acceso sino, en su lugar, instalar una caché web en la red institucional. Esta solución se ilustra en la Figura 2.13. Las tasas de acierto, es decir, la fracción de solicitudes que son satisfechas por una caché, suelen estar comprendidas, en la práctica, entre 0,2 y 0,7. Con propósitos ilustrativos, supongamos que la caché proporciona una tasa de acierto de 0,4 para esta institución. Dado que los clientes y la caché están conectados a la misma LAN de alta velocidad, el 40 por ciento de la solicitudes serán satisfechas casi de forma inmediata, es decir, en 10 milisegundos o menos por la caché. No obstante, el restante 60 por ciento de las solicitudes todavía tendrán que ser satisfechas por los servidores de origen. Pero sólo con el 60 por ciento de los objetos solicitados atravesando el enlace de acceso, la intensidad de tráfico en el mismo se reduce de 1,0 a 0,6. Típicamente, una intensidad de tráfico menor que 0,8 se corresponde con un retardo pequeño, digamos de unas decenas de milisegundos, en un enlace de 15 Mbps. Este retardo es despreciable comparado con los dos segundos del retardo de Internet. Teniendo en cuenta todo esto, el retardo medio será entonces:

$$0{,}4 \cdot (0{,}01 \text{ segundos}) + 0{,}6 \cdot (2{,}01 \text{ segundos})$$

lo que es algo más de 1,2 segundos. Por tanto, esta segunda solución proporciona incluso un tiempo de respuesta menor que la primera y no requiere que la institución actualice su enlace con Internet. Por supuesto, la institución tiene que comprar e instalar una caché web, pero este coste es bajo, ya que muchas cachés utilizan software de dominio público que se ejecuta en computadoras PC baratas.

2.2.6 GET condicional

Aunque el almacenamiento en caché puede reducir los tiempos de respuesta percibidos por el usuario, introduce un nuevo problema: la copia de un objeto que reside en la caché puede estar desactualizada. En otras palabras, el objeto almacenado en el servidor web puede haber sido modificado desde que la copia fue almacenada en la caché del cliente. Afortunadamente, HTTP dispone de un mecanismo que permite a la caché verificar que sus objetos están actualizados. Este mecanismo se denomina **GET condicional**. Un mensaje de solicitud HTTP se denomina también mensaje GET condicional si (1) el mensaje de solicitud utiliza el método `GET` y (2) el mensaje de solicitud incluye una línea de cabecera `If-Modified-Since:`.

Para ilustrar cómo opera el GET condicional, veamos un ejemplo. En primer lugar, una caché proxy envía un mensaje de solicitud a un servidor web en nombre de un navegador que realiza una solicitud:

```
GET /fruit/kiwi.gif HTTP/1.1
Host: www.exotiquecuisine.com
```

En segundo lugar, el servidor web envía a la caché un mensaje de respuesta con el objeto solicitado:

```
HTTP/1.1 200 OK
Date: Sat, 7 Jul 2007 15:39:29
Server: Apache/1.3.0 (Unix)
Last-Modified: Wed, 4 Jul 2007 09:23:24
Content-Type: image/gif

(datos datos datos datos datos ...)
```

La caché reenvía el objeto al navegador que lo ha solicitado pero también lo almacena localmente. Y lo que es más importante, la caché también almacena la fecha de la última modificación junto con el objeto. En tercer lugar, una semana después, otro navegador solicita el mismo objeto a través de la caché y el objeto todavía se encuentra almacenado allí. Puesto que este objeto puede haber sido modificado en el servidor web en el transcurso de la semana, la caché realiza una comprobación de actualización ejecutando un GET condicional. Específicamente, la caché envía:

```
GET /fruit/kiwi.gif HTTP/1.1
Host: www.exotiquecuisine.com
If-modified-since: Wed, 4 Jul 2007 09:23:24
```

Observe que el valor de la línea de cabecera `If-modified-since:` es exactamente igual al valor de la línea de cabecera `Last-Modified:` que fue enviada por el servidor hace una semana. Este GET condicional le pide al servidor que envíe el objeto sólo si éste ha sido modificado después de la última fecha especificada. Suponga que el objeto no ha sido modificado desde el 4 de julio de 2007 a las 09:23:24. Por último y en cuarto lugar, el servidor web envía un mensaje de respuesta a la caché:

```
HTTP/1.1 304 Not Modified
Date: Sat, 14 Jul 2007 15:39:29
Server: Apache/1.3.0 (Unix)

(cuerpo de entidad vacío)
```

Vemos que en respuesta al GET condicional el servidor web envía un mensaje de respuesta, pero no incluye el objeto solicitado en el mismo. Si incluyera el objeto solicitado sólo conseguiría desperdiciar ancho de banda e incrementar el tiempo de respuesta percibido por el usuario, especialmente si el tamaño del objeto es grande. Observe que este último mensaje de respuesta muestra en la línea de estado el texto `304 Not Modified` (no modificado), lo que indica a la caché que puede reenviar la copia del objeto que tiene en caché al navegador que lo haya solicitado.

Aquí terminamos nuestra exposición acerca de HTTP, el primer protocolo de Internet (un protocolo de la capa de aplicación) que hemos estudiado en detalle. Hemos examinado el formato de los mensajes HTTP y las acciones realizadas por el servidor y el cliente web según se envían y reciben estos mensajes. También hemos visto algo acerca de la infraestructura de las aplicaciones web, incluyendo las cachés, las cookies y las bases de datos back-end, elementos todos ellos que están ligados de alguna manera con el protocolo HTTP.

2.3 Transferencia de archivos: FTP

En una sesión FTP típica, el usuario está sentado frente a un host (el host local) y desea transferir archivos a o desde un host remoto. Para que el usuario pueda acceder a la cuenta remota, debe proporcionar una identificación de usuario y una contraseña. Una vez proporcionada esta información de autorización, el usuario puede transferir archivos desde el sistema de archivos local al sistema de archivos remoto, y viceversa. Como se muestra en la Figura 2.14, el usuario interactúa con FTP a través de un agente de usuario FTP. En primer lugar, el usuario proporciona el nombre del host remoto, lo que hace que el proceso cliente FTP en el host local establezca una conexión TCP con el proceso servidor FTP en el host remoto. A continuación, el usuario proporciona su identificación y su contraseña, que son enviados a través de la conexión TCP como parte de los comandos FTP. Una vez que el servidor ha autorizado al usuario, el usuario copia uno o más archivos almacenados en el sistema de archivos local en el sistema de archivos remoto (o viceversa).

HTTP y FTP son protocolos de transferencia de archivos y tienen muchas características en común; por ejemplo, ambos se ejecutan por encima de TCP. Sin embargo, estos dos protocolos de la capa de aplicación también presentan algunas diferencias importantes. La diferencia principal es que FTP utiliza dos conexiones TCP paralelas para transferir un archivo, una **conexión de control** y una **conexión de datos**. La conexión de control se emplea para enviar información de control entre los dos hosts, como la identificación del

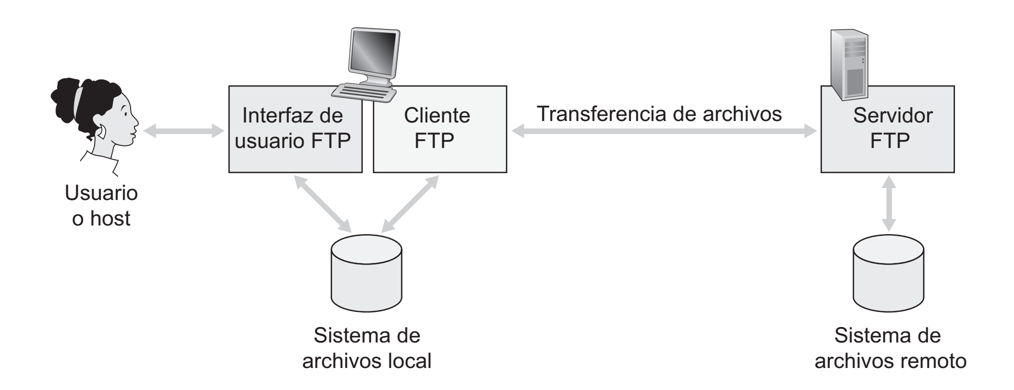


Figura 2.14 • FTP transfiere archivos entre un sistema de archivos local y un sistema de archivos remoto.

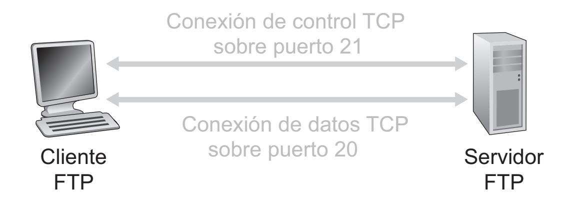


Figura 2.15 • Conexiones de control y de datos.

usuario, la contraseña, comandos para modificar el directorio remoto y comandos para "introducir" (PUT) y "extraer" (GET) archivos. La conexión de datos se utiliza para enviar un archivo. Puesto que FTP utiliza una conexión de control separada, se dice que FTP envía su información de control **fuera de banda**. En el Capítulo 7 veremos que el protocolo RTSP, que se utiliza para controlar la transferencia de medios continuos como audio y vídeo, también envía su información de control fuera de banda. Como recordará, HTTP envía las líneas de cabecera de solicitud y respuesta por la misma conexión TCP que transporta el propio archivo transferido. Por esta razón, se dice que HTTP envía su información de control **en banda**. En la siguiente sección veremos que SMTP, el principal protocolo para el correo electrónico, también envía la información de control en banda. En la Figura 2.15 se ilustran las conexiones de control y de datos de FTP.

Cuando un usuario inicia una sesión FTP con un host remoto, el lado del cliente de FTP (usuario) establece en primer lugar una conexión de control TCP con el lado del servidor (host remoto) en el puerto de servidor número 21. El lado del cliente de FTP envía la identificación y la contraseña del usuario a través de esta conexión de control. El lado del cliente FTP también envía, a través de la conexión control, comandos para modificar el directorio remoto. Cuando el lado del servidor recibe un comando para realizar una transferencia de archivo a través de la conexión de control (bien a, o desde el host remoto), el lado del servidor inicia una conexión de datos TCP con el lado del cliente. FTP envía exactamente un archivo a través de la conexión de datos y luego cierra la conexión de datos. Si durante la misma sesión el usuario desea transferir otro archivo, FTP abre otra conexión de datos. Luego, con FTP, la conexión de control permanece abierta mientras que

dure la sesión de usuario, pero se crea una nueva conexión de datos para cada archivo transferido dentro de la sesión (es decir, las conexiones de datos no son persistentes).

A lo largo de la sesión, el servidor FTP tiene que mantener un **estado** del usuario. En concreto, el servidor debe asociar la conexión de control con una cuenta de usuario específica y debe estar al tanto del directorio actual del usuario cuando éste se mueve por el árbol del directorio remoto. Llevar el control de esta información de estado para cada sesión de usuario activa restringe de forma significativa el número total de sesiones que FTP puede mantener simultáneamente. Recuerde que HTTP, por el contrario, es un protocolo sin memoria del estado (no tiene que recordar el estado de los usuarios).

2.3.1 Comandos y respuestas de FTP

Vamos a terminar esta sección con una breve explicación de algunos de los comandos y respuestas de FTP más comunes. Los comandos, del cliente al servidor, y las respuestas, del servidor al cliente, se envían a través de la conexión de control en formato ASCII de 7 bits. Por tanto, al igual que los comandos HTTP, las personas pueden leer los comandos FTP. Para separar los comandos sucesivos se añade un retorno de carro y un salto de línea al final de cada comando. Cada comando consta de cuatro caracteres ASCII en letras mayúsculas y algunos emplean argumentos opcionales. Algunos de los comandos más comunes son los siguientes:

- `USER nombre_de_usuario`: se utiliza para enviar la identificación del usuario al servidor.
- `PASS contraseña`: se utiliza para enviar la contraseña del usuario al servidor.
- `LIST`: se utiliza para pedir al servidor que devuelva una lista de todos los archivos existentes en el directorio remoto actual. La lista de archivos se envía a través de una conexión de datos (nueva y no persistente), en lugar de a través de la conexión de control TCP.
- `RETR nombre_de_archivo`: se utiliza para recuperar (es decir, extraer) un archivo del directorio actual del host remoto. Este comando hace que el host remoto inicie una conexión de datos y envíe el archivo solicitado a través de la conexión de datos.
- `STOR nombre_de_archivo`: se utiliza para almacenar (es decir, introducir) un archivo en el directorio actual del host remoto.

Existe una correspondencia uno a uno entre el comando que ejecuta el usuario y el comando FTP enviado a través de la conexión de control. Cada comando va seguido de una respuesta, enviada desde el servidor al cliente. Las respuestas son números de tres dígitos, con un mensaje opcional que sigue al número. Esta estructura es similar a la del código de estado y la frase explicativa de la línea de estado en los mensajes de respuesta HTTP. Algunas respuestas típicas, junto con sus posibles frases, son las siguientes:

- `331 Username OK, password required`
- `125 Data connection already open; transfer starting`
- `425 Can't open data connection`
- `452 Error writing file`

Los lectores que estén interesados en obtener más información acerca de otros comandos FTP y sus respuestas les animamos a leer el documento RFC 959.

TP 6

2.4 Correo electrónico en Internet

El correo electrónico ha existido desde que Internet viera la luz. Era la aplicación más popular cuando Internet estaba en sus comienzos [Segaller 1998] y a lo largo de los años se ha ido convirtiendo en una aplicación cada vez más elaborada y potente. En la actualidad continúa siendo una de las aplicaciones más importantes y utilizadas de Internet.

Al igual que el servicio ordinario de correo postal, el correo electrónico es un medio de comunicación asíncrono (las personas envían y leen los mensajes cuando les conviene, sin tener que coordinarse con las agendas de otras personas). En contraste con el correo postal, el correo electrónico es rápido, fácil de distribuir y barato. Las aplicaciones modernas de correo electrónico disponen de muchas características potentes. Mediante las listas de correo, se pueden enviar mensajes de correo electrónico y correo basura (*spam*) a miles de destinatarios a un mismo tiempo. A menudo, los mensajes de correo electrónico incluyen adjuntos, hipervínculos, texto en formato HTML y fotografías.

En esta sección vamos a examinar los protocolos de la capa de aplicación que forman la base del correo electrónico en Internet. Pero antes de sumergirnos en una explicación detallada acerca de estos protocolos, vamos a proporcionar una visión de conjunto del sistema de correo de Internet y sus componentes fundamentales.

La Figura 2.16 muestra una visión general del sistema de correo de Internet. A partir de este diagrama, vemos que existen tres componentes principales: **agentes de usuario**, **servidores de correo** y el **Protocolo simple de transferencia de correo (SMTP, *Simple Mail Transfer Protocol*)**. A continuación vamos a describir cada uno de estos componentes en el contexto de un emisor, Alicia, que envía un mensaje de correo electrónico a un destinatario, Benito. Los agentes de usuario permiten a los usuarios leer, responder, reenviar, guardar y componer mensajes. (Los agentes de usuario para el correo electrónico a veces se denominan *lectores de correo,* aunque en general nosotros hemos evitado el uso de este término a lo largo del libro.) Cuando Alicia termina de componer su mensaje, su agente de usuario envía el mensaje a su servidor de correo, donde el mensaje es colocado en la cola de mensajes salientes del servidor de correo. Cuando Benito quiere leer un mensaje, su agente de usuario recupera el mensaje de su buzón, que se encuentra en el servidor de correo. A finales de la década de 1990 comenzaron a popularizarse los agentes de usuario con interfaz gráfica de usuario (GUI, *Graphical User Interface*), permitiendo a los usuarios ver y componer mensajes multimedia. Actualmente, Microsoft Outlook, Apple Mail y Mozilla Thunderbird se encuentran entre los agentes de usuario con interfaz GUI más populares para correo electrónico. Existen también muchas interfaces de usuario de dominio público para correo electrónico basadas en texto (como por ejemplo, mail, pine y elm), así como interfaces basadas en la Web, como veremos enseguida.

Los servidores de correo forman el núcleo de la infraestructura del correo electrónico. Cada destinatario, como por ejemplo Benito, tiene un **buzón de correo** ubicado en uno de los servidores de correo. El buzón de Benito gestiona y mantiene los mensajes que le han sido enviados. Un mensaje típico inicia su viaje en el agente de usuario del emisor, viaja hasta el servidor de correo del emisor y luego hasta el servidor de correo del destinatario, donde es depositado en el buzón del mismo. Cuando Benito quiere acceder a los mensajes contenidos en su buzón, el servidor de correo que lo contiene autentica a Benito (mediante el nombre de usuario y la contraseña). El servidor de correo de Alicia también tiene que ocuparse de los fallos que se producen en el servidor de correo de Benito. Si el servidor de Alicia no puede enviar el mensaje de correo al servidor de Benito, entonces el servidor de Alicia

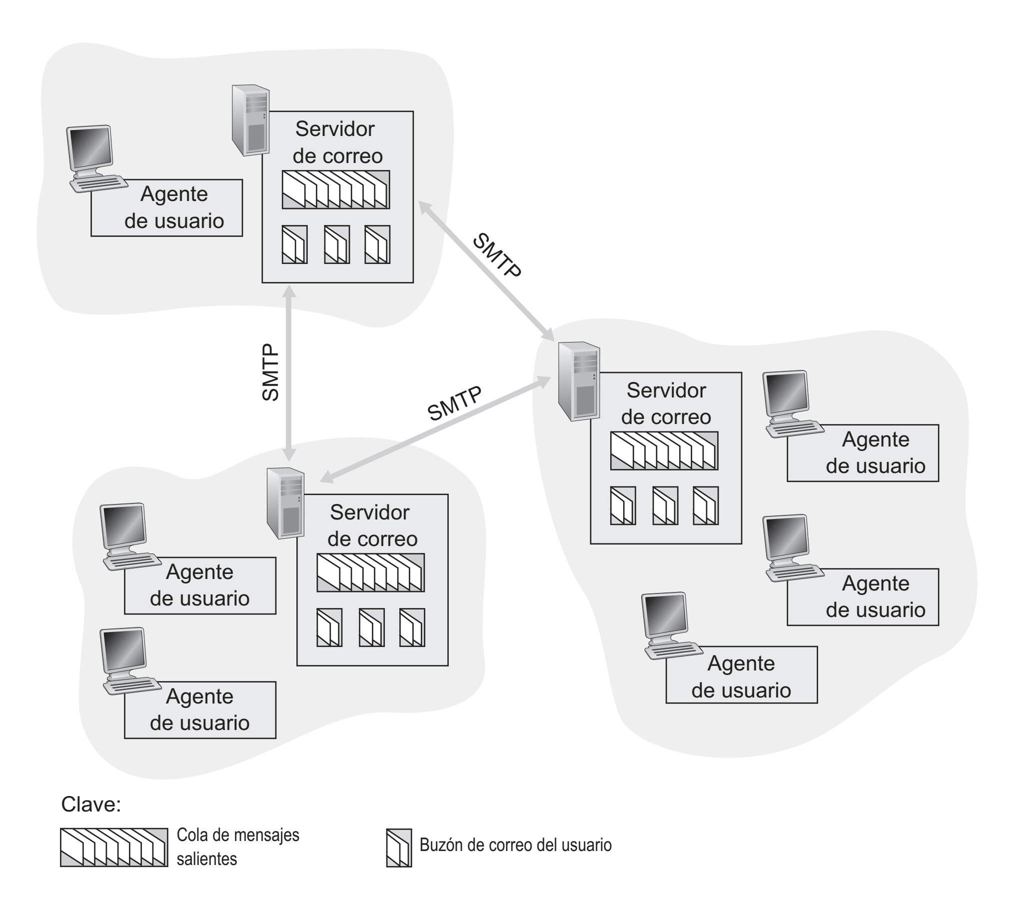


Figura 2.16 • Esquema general del sistema de correo electrónico de Internet.

mantiene el mensaje en una **cola de mensajes** e intenta enviarlo más tarde. Normalmente, los reintentos de envío se realizan más o menos cada 30 minutos; si después de varios días no se ha conseguido, el servidor elimina el mensaje y se lo notifica al emisor (Alicia) mediante un mensaje de correo electrónico.

SMTP es el principal protocolo de la capa de aplicación para el correo electrónico por Internet. Utiliza el servicio de transferencia de datos fiable de TCP para transferir el correo desde el servidor de correo del emisor al servidor de correo del destinatario. Al igual que la mayoría de los protocolos de la capa de aplicación, SMTP tiene dos lados: el lado del cliente, que se ejecuta en el servidor de correo del emisor, y el lado del servidor, que se ejecuta en el servidor de correo del destinatario. Tanto el lado del cliente como el del servidor de SMTP se ejecutan en todos los servidores de correo. Cuando un servidor de correo envía mensajes de correo a otros servidores de correo, actúa como un cliente SMTP. Cuando un servidor de correo recibe correo de otros servidores, actúa como un servidor SMTP.

2.4.1 SMTP

El protocolo SMTP, que está definido en el documento RFC 5321, es el corazón del correo electrónico por Internet. Como hemos mencionado anteriormente, SMTP transfiere mensa-



HISTORIA

CORREO ELECTRÓNICO WEB

En diciembre de 1995, sólo unos pocos años después de que se "inventara" la Web, Sabeer Bhatia y Jack Smith visitaron la empresa de capital riesgo en Internet de Draper Fisher Jurvetson y le propusieron desarrollar un sistema gratuito de correo electrónico basado en la Web. La idea era proporcionar una cuenta de correo electrónico gratuita a cualquiera que lo deseara y hacer que las cuentas fueran accesibles desde la Web. A cambio de un 15 por ciento de la compañía, Draper Fisher Jurvetson financió a Bhatia y Smith, que montaron la empresa Hotmail. Con tres personas trabajando a tiempo completo y 14 a tiempo parcial, que trabajaban a cambio de opción de compra de acciones, fueron capaces de desarrollar y lanzar el servicio en julio de 1996. Un mes después del lanzamiento tenían 100.000 suscriptores. En diciembre de 1997, menos de 18 meses después del lanzamiento del servicio, Hotmail tenía aproximadamente 12 millones de suscriptores y fue adquirida por Microsoft por 400 millones de dólares. El éxito de Hotmail a menudo se atribuye a su "ventaja de ser el primero en tomar la iniciativa" y al "marketing viral" intrínseco al correo electrónico. (Quizá algunos de los estudiantes que estén leyendo este libro se encuentren entre los nuevos emprendedores que serán los primeros en concebir y desarrollar servicios de Internet con "ventaja de ser el primero" y con técnicas inherentes de "marketing viral".)

El correo electrónico continúa proliferando, haciéndose cada año que pasa más sofisticado y potente. Uno de los servicios más populares actualmente es gmail de Google, que ofrece gigabytes de almacenamiento gratuito, servicios avanzados de filtrado de correo basura y de detección de virus, un servicio opcional de cifrado de correo electrónico (con SSL), acceso a servicios de correo electrónico de terceros y una interfaz orientada a la realizacion de búsquedas.

jes desde los servidores de correo de los emisores a los servidores de correo de los destinatarios. SMTP es mucho más antiguo que HTTP. (El RFC original que se ocupa de SMTP data de 1982 y SMTP ya existía bastante tiempo antes.) Aunque SMTP tienen muchas cualidades maravillosas, como prueba su presencia en Internet, es una tecnología heredada que utiliza algunas funcionalidades arcaicas. Por ejemplo, restringe el cuerpo (no sólo las cabeceras) de todos los mensajes a formato ASCII de 7 bits. Esta restricción tenía sentido a principios de la década de 1980, cuando la capacidad de transmisión era escasa y nadie enviaba mensajes con adjuntos o imágenes de gran tamaño, o archivos de audio o vídeo. Pero actualmente, en la era multimedia, la restricción del formato ASCII de 7 bits causa muchos problemas: requiere que los datos binarios multimedia se codifiquen a ASCII antes de ser transmitidos a través de SMTP y requiere que el correspondiente mensaje ASCII sea decodificado de vuelta a binario una vez realizado el transporte SMTP. Recuerde, como hemos visto en la Sección 2.2, que HTTP no precisa que los datos multimedia sean codificados a ASCII antes de ser transferidos.

Para ilustrar el funcionamiento básico de SMTP vamos a recurrir a un escenario ya conocido. Suponga que Alicia desea enviar a Benito un sencillo mensaje ASCII.

1. Alicia invoca a su agente de usuario para correo electrónico, proporciona la dirección de correo electrónico de Benito (por ejemplo, `benito@unaescuela.edu`), compone un mensaje e indica al agente de usuario que lo envíe.
2. El agente de usuario de Alicia envía el mensaje al servidor de correo de ella, donde es colocado en una cola de mensajes.
3. El lado del cliente de SMTP, que se ejecuta en el servidor de correo de Alicia, ve el mensaje en la cola de mensajes. Abre una conexión TCP con un servidor SMTP, que se ejecuta en el servidor de correo de Benito.
4. Después de la fase de negociación inicial de SMTP, el cliente SMTP envía el mensaje de Alicia a través de la conexión TCP.
5. En el servidor de correo de Benito, el lado del servidor de SMTP recibe el mensaje. El servidor de correo de Benito coloca entonces el mensaje en el buzón de Benito.
6. Benito invoca a su agente de usuario para leer el mensaje cuando le apetezca.

En la Figura 2.17 se resume este escenario.

Es importante observar que normalmente SMTP no utiliza servidores de correo intermedios para enviar correo, incluso cuando los dos servidores de correo se encuentran en extremos opuestos del mundo. Si el servidor de Alicia está en Hong Kong y el de Benito está en St. Louis, la conexión TCP será una conexión directa entre los servidores de Hong Kong y St. Louis. En particular, si el servidor de correo de Benito está fuera de servicio, el servidor de Alicia conservará el mensaje y lo intentará de nuevo (el mensaje no se deja en un servidor de correo intermedio).

Veamos en detalle cómo transfiere SMTP un mensaje desde un servidor de correo emisor a un servidor de correo receptor. Comprobaremos que el protocolo SMTP presenta muchas similitudes con los protocolos empleados por las personas para las interacciones cara a cara. En primer lugar, el cliente SMTP (que se ejecuta en el host servidor de correo emisor) establece una conexión TCP con el puerto 25 del servidor SMTP (que se ejecuta en el host servidor de correo receptor). Si el servidor no está operativo, el cliente lo intentará más tarde. Una vez que se ha establecido la conexión, el servidor y el cliente llevan a cabo el proceso de negociación de la capa de aplicación (al igual que las personas, que antes de intercambiar información se presentan, los clientes y servidores SMTP se presentan a sí mismos antes de transferir la información). Durante esta fase de negociación SMTP, el cliente

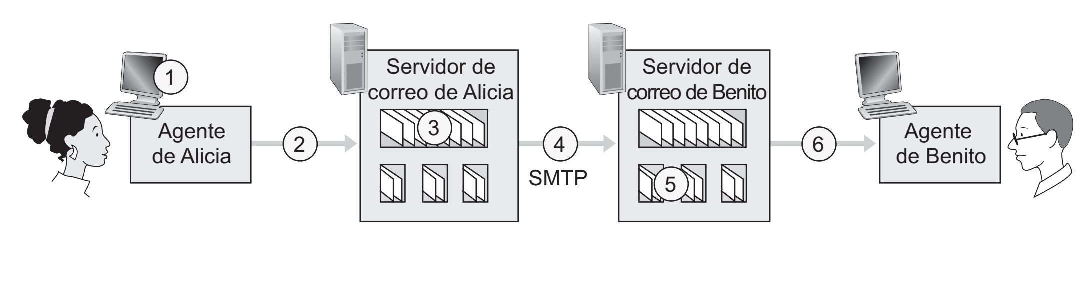


Figura 2.17 • Alicia envía un mensaje a Benito.

SMTP especifica la dirección de correo electrónico del emisor (la persona que ha generado el mensaje) y la dirección de correo electrónico del destinatario. Una vez que el cliente y el servidor SMTP se han presentado a sí mismos, el cliente envía el mensaje. SMTP cuenta con el servicio de transferencia de datos fiable de TCP para transferir el mensaje al servidor sin errores. El cliente repite entonces este proceso a través de la misma conexión TCP si tiene que enviar otros mensajes al servidor; en caso contrario, indica a TCP que cierre la conexión.

Veamos ahora una transcripción de ejemplo de los mensajes intercambiados entre un cliente SMTP (C) y un servidor SMTP (S). El nombre de host del cliente es crepes.fr y el nombre de host del servidor es hamburger.edu. Las líneas de texto ASCII precedidas por C: son exactamente las líneas que el cliente envía a su socket TCP y las líneas de texto ASCII precedidas por S: son las líneas que el servidor envía a su socket TCP. La siguiente transcripción comienza tan pronto como se establece la conexión TCP.

```
S:    220 hamburger.edu
C:    HELO crepes.fr
S:    250 Hello crepes.fr, pleased to meet you
C:    MAIL FROM: <alicia@crepes.fr>
S:    250 alicia@crepes.fr ... Sender ok
C:    RCPT TO: <benito@hamburger.edu>
S:    250 benito@hamburger.edu ... Recipient ok
C:    DATA
S:    354 Enter mail, end with "." on a line by itself
C:    ¿Te gusta el ketchup?
C:    ¿Y los pepinillos en vinagre?
C:    .
S:    250 Message accepted for delivery
C:    QUIT
S:    221 hamburger.edu closing connection
```

En el ejemplo anterior, el cliente envía un mensaje ("¿Te gusta el ketchup? ¿Y los pepinillos en vinagre?") desde el servidor de correo crepes.fr al servidor de correo hamburger.edu. Como parte del diálogo, el cliente ejecuta cinco comandos: HELO (una abreviatura de HELLO), MAIL FROM, RCPT TO, DATA y QUIT. Estos comandos se explican por sí mismos. El cliente también envía una línea que consta únicamente de un punto, que indica el final del mensaje para el servidor. (En la jerga ASCII, cada mensaje termina con CRLF.CRLF, donde CR y LF se corresponden con el retorno de carro y el salto de línea, respectivamente.) El servidor responde a cada comando, teniendo cada una de las respuestas un código de respuesta y una explicación (opcional) en inglés. Hemos mencionado que SMTP utiliza conexiones persistentes: si el servidor de correo emisor tiene varios mensajes que enviar al mismo servidor de correo receptor, puede enviar todos los mensajes a través de la misma conexión TCP. Para cada mensaje, el cliente inicia el proceso con un nuevo comando MAIL FROM: crepes.fr, designa el final del mensaje con un único punto y ejecuta el comando QUIT sólo después de que todos los mensajes hayan sido enviados.

Es muy recomendable que utilice Telnet para establecer un diálogo directo con un servidor SMTP. Para ello, ejecute:

```
telnet NombreServidor 25
```

donde `NombreServidor` es el nombre de un servidor de correo local. Al hacer esto, simplemente está estableciendo una conexión TCP entre su host local y el servidor de correo. Después de escribir esta línea, debería recibir inmediatamente la respuesta `220` del servidor. A continuación, ejecute los comandos SMTP `HELO`, `MAIL FROM`, `RCPT TO`, `DATA`, `CRLF.CRLF` y `QUIT` en los instantes apropiados. También es extremadamente recomendable que realice la Tarea de programación 2 incluida al final del capítulo. En esta tarea, tendrá que crear un agente de usuario simple que implemente el lado del cliente de SMTP. Esto le permitirá enviar un mensaje de correo electrónico a un destinatario arbitrario a través de un servidor de correo local.

2.4.2 Comparación con HTTP

Vamos ahora a comparar brevemente SMTP con HTTP. Ambos protocolos se emplean para transferir archivos de un host a otro: HTTP transfiere archivos (también denominados objetos) desde un servidor web a un cliente web (normalmente, un navegador); SMTP transfiere archivos (es decir, mensajes de correo electrónico) desde un servidor de correo a otro servidor de correo. Para transferir los archivos, tanto HTTP persistente como SMTP emplean conexiones persistentes. Por tanto, ambos protocolos tienen características comunes. Sin embargo, también presentan algunas diferencias. En primer lugar, HTTP es principalmente un **protocolo pull** (protocolo de extracción): alguien carga la información en un servidor web y los usuarios utilizan HTTP para extraer la información del servidor cuando desean. En concreto, la máquina que desea recibir el archivo inicia la conexión TCP. Por el contrario, SMTP es fundamentalmente un **protocolo push** (protocolo de inserción): el servidor de correo emisor introduce el archivo en el servidor de correo receptor. En concreto, la máquina que desea enviar el archivo inicia la conexión TCP.

Una segunda diferencia, a la que hemos aludido anteriormente, es que SMTP requiere que cada mensaje, incluyendo el cuerpo de cada mensaje, esté en el formato ASCII de 7 bits. Si el mensaje contiene caracteres que no corresponden a dicho formato (como por ejemplo, caracteres acentuados) o contiene datos binarios (como por ejemplo un archivo de imagen), entonces el mensaje tiene que ser codificado en ASCII de 7 bits. Los datos HTTP no imponen esta restricción.

Una tercera diferencia importante tiene que ver con cómo se maneja un documento que conste de texto e imágenes (junto con posiblemente otros tipos multimedia). Como hemos visto en la Sección 2.2, HTTP encapsula cada objeto en su propio mensaje de respuesta HTTP. El correo Internet incluye todos los objetos del mensaje en un mismo mensaje.

2.4.3 Formatos de los mensajes de correo

Cuando Alicia escribe una carta por correo ordinario a Benito, puede incluir todo tipo de información accesoria en la parte superior de la carta, como la dirección de Benito, su propia dirección de respuesta y la fecha. De forma similar, cuando una persona envía un mensaje de correo electrónico a otra, una cabecera que contiene la información administrativa antecede al cuerpo del mensaje. Esta información se incluye en una serie de líneas de cabecera, que están definidas en el documento RFC 5322. Las líneas de cabecera y el cuerpo del mensaje se separan mediante una línea en blanco (es decir, mediante `CRLF`). RFC 5322 especifica el formato exacto de las líneas de cabecera, así como sus interpretaciones semánticas. Como con HTTP, cada línea de cabecera contiene texto legible, que

consta de una palabra clave seguida de dos puntos y de un valor. Algunas de las palabras clave son obligatorias y otras son opcionales. Toda cabecera tiene que estar formada por una línea de cabecera `From:` y una línea de cabecera `To:`; también puede incluir una línea `Subject:`, así como otras líneas de cabecera opcionales. Es importante destacar que estas líneas de cabecera son *diferentes* de los comandos SMTP que hemos estudiado en la Sección 2.4.1 (incluso aunque contengan algunas palabra comunes como "*from*" y "*to*"). Los comandos vistos en esa sección forman parte del protocolo de negociación de SMTP; las líneas de cabecera examinadas en esta sección forman parte del propio mensaje de correo.

Una cabecera de mensaje típica sería como la siguiente:

```
From: alicia@crepes.fr
To: benito@hamburger.edu
Subject: Búsqueda del significado de la vida.
```

Después del mensaje de cabecera se incluye una línea en blanco y, a continuación, el cuerpo del mensaje (en ASCII). Debería utilizar Telnet para enviar un mensaje a un servidor de correo que contenga varias líneas de cabecera, incluyendo la línea de cabecera del asunto `Subject:`. Para ello, ejecute el comando `telnet NombreServidor 25`, como se ha explicado en la Sección 2.4.1.

2.4.4 Protocolos de acceso para correo electrónico

Una vez que SMTP envía el mensaje del servidor de correo de Alicia al servidor de correo de Benito, el mensaje se coloca en el buzón de este último. A lo largo de esta exposición, hemos supuesto tácitamente que Benito lee su correo registrándose en el host servidor y utilizando después un lector de correo que se ejecuta en dicho host. Hasta principios de la década de 1990 ésta era la forma habitual de hacer las cosas. Pero, actualmente, el acceso al correo electrónico utiliza una arquitectura cliente-servidor; el usuario típico lee el correo electrónico con un cliente que se ejecuta en el sistema terminal del usuario, por ejemplo, en un PC de la oficina, en un portátil o en una PDA. Ejecutando un cliente de correo en un PC local, los usuarios disponen de un rico conjunto de funcionalidades, entre las que se incluye la posibilidad de visualizar documentos adjuntos y mensajes multimedia.

Puesto que Benito (el destinatario) ejecuta su agente de usuario en su PC local, es natural que considere también incluir un servidor de correo en su PC local. De esta forma, el servidor de correo de Alicia dialogará directamente con el PC de Benito. Sin embargo, hay un problema con este método. Recuerde que un servidor de correo gestiona buzones de correo y ejecuta los lados del cliente y del servidor de SMTP. Si el servidor de correo de Benito residiera en su PC local, entonces el PC de Benito tendría que estar siempre encendido y conectado a Internet para recibir los nuevos correos que pudieran llegar en cualquier momento. Pero esto es impracticable para muchos usuarios de Internet. En su lugar, un usuario típico ejecuta un agente de usuario en el PC local pero accede a su buzón de correo almacenado en un servidor de correo compartido que siempre está encendido. Este servidor es compartido con otros usuarios y, normalmente, mantenido por el ISP del usuario (por ejemplo, una universidad o una empresa).

Ahora vamos a ver qué ruta sigue un mensaje de correo electrónico que Alicia envía a Benito. Acabamos de estudiar que en algún punto a lo largo de la ruta el mensaje de correo electrónico tiene que ser depositado en el servidor de correo de Benito. Esto se podría conseguir fácilmente haciendo que el agente de usuario de Alicia envíe el mensaje directamente

al servidor de correo de Benito. Y esto podría hacerse utilizando SMTP (de hecho, SMTP ha sido diseñado para llevar el correo electrónico de un host a otro. Sin embargo, normalmente el agente de usuario del emisor no se comunica directamente con el servidor de correo del destinatario. En su lugar, como se muestra en la Figura 2.18, el agente de usuario de Alicia utiliza SMTP para introducir el mensaje de correo en su propio servidor de correo, y a continuación el servidor de correo de Alicia utiliza SMTP (como un cliente SMTP) para pasar el mensaje al servidor de correo de Benito. ¿Por qué este procedimiento en dos pasos? Fundamentalmente, porque sin la retransmisión a través del servidor de correo de Alicia, el agente de usuario de ésta no tiene ninguna forma de acceder a un servidor de correo de destino inalcanzable. Al hacer que Alicia deposite primero el mensaje en su propio servidor de correo, éste puede intentar una y otra vez enviar el mensaje al servidor de correo de Benito, por ejemplo, cada 30 minutos, hasta que el servidor de Benito esté de nuevo operativo. (Y si el servidor de correo de Alicia no funciona, entonces ella tiene el recurso de ¡quejarse al administrador del sistema!). El RFC que se ocupa de SMTP define cómo se pueden utilizar los comandos SMTP para transmitir un mensaje a través de varios servidores SMTP.

¡Pero todavía falta una pieza del puzzle! ¿Cómo un destinatario como Benito, que ejecuta un agente de usuario en su PC local, obtiene sus mensajes, que se encuentran en un servidor de correo de su ISP? Tenga en cuenta que el agente de usuario de Benito no puede utilizar SMTP para obtener los mensajes porque es una operación de extracción (pull) mientras que SMTP es un protocolo push (de inserción). Así, el puzzle se completa añadiendo un protocolo especial de acceso al correo que permita transferir los mensajes del servidor de correo de Benito a su PC local. Actualmente existen varios protocolos de acceso a correo electrónico populares, entre los que se incluyen el **Protocolo de oficina de correos versión 3 (POP3,** ***Post Office Protocol—Version 3*****)**, el **Protocolo de acceso de correo de Internet (IMAP,** ***Internet Mail Access Protocol*****)** y HTTP.

La Figura 2.18 proporciona un resumen de los protocolos que se utilizan para el correo de Internet: SMTP se emplea para transferir correo desde el servidor de correo del emisor al servidor de correo del destinatario; SMTP también se utiliza para transferir correo desde el agente de usuario del emisor al servidor de correo del mismo. Para transferir los mensajes de correo almacenados en el servidor de correo del destinatario al agente de usuario del mismo se emplea un protocolo de acceso a correo, como POP3.

POP3

POP3 es un protocolo de acceso a correo extremadamente simple. Está definido en [RFC 1939], que es un documento corto y bastante claro. Dado que el protocolo es tan simple, su funcionalidad es bastante limitada. POP3 se inicia cuando el agente de usuario (el

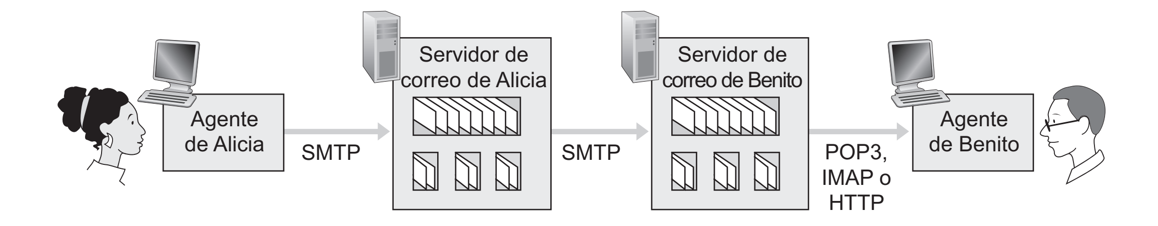


Figura 2.18 • Protocolos de correo electrónico y entidades que los utilizan.

cliente) abre una conexión TCP en el puerto 110 al servidor de correo (el servidor). Una vez establecida la conexión TCP, POP3 pasa a través de tres fases: autorización, transacción y actualización. Durante la primera fase, la autorización, el agente de usuario envía un nombre de usuario y una contraseña (en texto legible) para autenticar al usuario. Durante la segunda fase, la de transacción, el agente de usuario recupera los mensajes; también durante esta fase, el agente de usuario puede marcar los mensajes para borrado, eliminar las marcas de borrado y obtener estadísticas de correo. La tercera fase, la actualización, tiene lugar después que el cliente haya ejecutado el comando `quit`, terminando la sesión POP3; en este instante, el servidor de correo borra los mensajes que han sido marcados para borrado.

En una transacción POP3, el agente de usuario ejecuta comandos y el servidor devuelve para cada comando una respuesta. Existen dos posibles respuestas: `+OK` (seguida en ocasiones por una serie de datos servidor-cliente), utilizada por el servidor para indicar que el comando anterior era correcto; y `-ERR`, utilizada por el servidor para indicar que había algún error en el comando anterior.

La fase de autorización tiene dos comandos principales: `user <nombreusuario>` y `pass <contraseña>`. Para ilustrar estos dos comandos, le sugerimos que establezca una conexión Telnet directamente en un servidor POP3, utilizando el puerto 110, y ejecute estos comandos. Suponga que `mailServer` es el nombre de su servidor de correo. Verá algo similar a lo siguiente:

```
telnet mailServer 110
+OK POP3 server ready
user benito
+OK
pass hambre
+OK user successfully logged on
```

Si escribe mal un comando, el servidor POP3 le responderá con un mensaje `-ERR`.

Abordemos ahora la fase de transacción. Un agente de usuario que utilice POP3 suele ser configurado (por el usuario) para “descargar y borrar” o para “descargar y guardar”. La secuencia de comandos que ejecute un agente de usuario POP3 dependerá de en cuál de estos dos modos esté operando. En el modo descargar y borrar, el agente de usuario ejecutará los comandos `list`, `retr` y `dele`. Por ejemplo, suponga que el usuario tiene dos mensajes en su buzón de correo. En el diálogo que proporcionamos a continuación, `C:` (que quiere decir cliente) es el agente de usuario y `S:` (que quiere decir servidor) es el servidor de correo. La transacción será similar a lo siguiente:

```
C: list
S: 1 498
S: 2 912
S: .
C: retr 1
S: (bla bla ...
S: .................
S: .........bla)
S: .
C: dele 1
```

```
C: retr 2
S: (bla bla ...
S: .................
S: ..........bla)
S: .
C: dele 2
C: quit
S: +OK POP3 server signing off
```

En primer lugar, el agente de usuario pide al servidor de correo que le informe del tamaño de cada uno de los mensajes almacenados. A continuación, el agente de usuario recupera y borra cada uno de los mensajes del servidor. Observe que después de la fase de autorización, el agente de usuario sólo ha utilizado cuatro comandos: `list`, `retr`, `dele` y `quit`. La sintaxis de estos comandos está definida en el documento RFC 1939. Después de procesar el comando `quit`, el servidor POP3 entra en la fase de actualización y borra los mensajes 1 y 2 del buzón.

Un problema en este modo de descarga y borrado es que el destinatario, Benito, puede ser algo nómada y puede desear poder acceder a sus mensajes de correo desde varias máquinas; por ejemplo, desde el PC de la oficina, el PC de su casa y el portátil. El modo de descarga y borrado reparte los mensajes de correo de Benito entre estas tres máquinas; en particular, si Benito lee un mensaje en el PC de la oficina, no podrá volver a leerlo en el portátil en su casa por la noche. En el modo descargar y guardar, el agente de usuario deja los mensajes en el servidor de correo después de que se hayan descargado. En este caso, Benito podrá volver a leer los mensajes desde distintas máquinas; puede acceder a un mensaje en el trabajo y luego también en su casa si lo desea.

Durante una sesión POP3 entre un agente de usuario y el servidor de correo, el servidor POP3 mantiene cierta información de estado; en concreto, mantiene la relación de los mensajes de usuario que han sido marcados para ser borrados. Sin embargo, el servidor POP3 no conserva la información de estado de una sesión POP3 a otra. Esta falta de memoria del estado entre sesiones simplifica enormemente la implementación de un servidor POP3.

IMAP

Con el acceso POP3, una vez que Benito ha descargado sus mensajes en la máquina local, puede crear carpetas de correo y mover los mensajes descargados a las carpetas. A continuación, puede borrar los mensajes, pasarlos a carpetas y realizar búsquedas de mensajes (por nombre del emisor o asunto). Pero este paradigma, es decir, las carpetas y los mensajes guardos en la máquina local, plantea un problema para el usuario nómada, que preferirá mantener una jerarquía de carpetas en un servidor remoto al que pueda acceder desde cualquier computadora. Esto no es posible con POP3 (el protocolo POP3 no proporciona ningún medio al usuario para crear carpetas remotas y asignar mensajes a las mismas).

Para resolver este y otros problemas se inventó el protocolo IMAP, definido en [RFC 3501]. Al igual que POP3, IMAP es un protocolo de acceso a correo. Ofrece muchas más funcionalidades que POP3, pero también es significativamente más complejo (y, por tanto, las implementaciones del lado del cliente y del lado del servidor son bastante más complejas).

Un servidor IMAP asociará cada mensaje con una carpeta; cuando un mensaje llega al servidor, se asocia con la carpeta INBOX (Bandeja de entrada) del destinatario, el cual puede entonces pasar el mensaje a una nueva carpeta creada por el usuario, leer el mensaje,

borrarlo, etc. El protocolo IMAP proporciona comandos que permiten a los usuarios crear carpetas y mover los mensajes de una carpeta a otra. IMAP también proporciona comandos que permiten a los usuarios realizar búsquedas en carpetas remotas para localizar mensajes que cumplan unos determinados criterios. Observe que, a diferencia de POP3, un servidor IMAP mantiene información acerca del estado a lo largo de las sesiones IMAP, como por ejemplo, los nombres de las carpetas y los mensajes asociados con cada una de ellas.

Otra importante característica de IMAP es que dispone de comandos que permiten a un agente de usuario obtener partes componentes de los mensajes. Por ejemplo, un agente de usuario puede obtener sólo la cabecera del mensaje o sólo una parte de un mensaje MIME de varias partes. Esta característica resulta útil cuando se emplea una conexión con un ancho de banda pequeño (por ejemplo, un enlace de módem de baja velocidad) entre el agente de usuario y su servidor de correo. Con una conexión de ancho de banda pequeño, puede que el usuario no quiera descargar todos los mensajes guardados en su buzón y que desee saltarse los mensajes largos que pueda haber; por ejemplo un archivo de audio o de vídeo. Puede obtener más información acerca de IMAP en el sitio oficial de IMAP [IMAP 2009].

Correo electrónico Web

Actualmente, cada vez más usuarios envían y acceden a su correo electrónico a través de sus navegadores web. Hotmail introdujo a mediados de la década de 1990 el acceso basado en la Web; actualmente, también ofrece este servicio Yahoo, Google, así como casi todas las principales universidades y empresas. Con este servicio, el agente de usuario es un navegador web corriente y el usuario se comunica con su buzón remoto a través de HTTP. Cuando un destinatario, como Benito, desea acceder a un mensaje de su buzón, éste es enviado desde el servidor de correo de Benito al navegador del mismo utilizando el protocolo HTTP en lugar de los protocolos POP3 o IMAP. Cuando un emisor, como Alicia, desea enviar un mensaje de correo electrónico, éste es transmitido desde su navegador a su servidor de correo a través de HTTP en lugar de mediante SMTP. Sin embargo, el servidor de correo de Alicia, continúa enviando mensajes a, y recibiendo mensajes de, otros servidores de correo que emplean SMTP.

TP7



2.5 DNS: servicio de directorio de Internet

Las personas podemos ser identificadas de muchas maneras. Por ejemplo, podemos ser identificadas por el nombre que aparece en nuestro certificado de nacimiento, por nuestro número de la seguridad social o por el número del carnet de conducir. Aunque cada uno de estos datos se puede utilizar para identificar a las personas, dentro de un determinado contexto un identificador puede ser más apropiado que otro. Por ejemplo, las computadoras del IRS (la terrible agencia tributaria de Estados Unidos) prefieren utilizar los números de la seguridad social de longitud fija que el nombre que aparece en el certificado de nacimiento. Por otro lado, las personas prefieren emplear el nombre que figura en los certificados de nacimiento, más fáciles de recordar, a los números de la seguridad social (puede imaginar que alguien le dijera "Hola. Mi nombre es 132-67-9875. Éste es mi marido, 178-87-1146.")

No sólo las personas podemos ser identificadas de diversas formas, los hosts de Internet también. Un identificador para los hosts es el **nombre de host**. Los nombres de host, como por ejemplo `cnn.com`, `www.yahoo.com`, `gaia.cs.umass.edu` y `cis.poly.edu`, son

mnemónicos y son, por tanto, entendidos por las personas. Sin embargo, los nombres de host proporcionan poca o ninguna información acerca de la ubicación del host dentro de Internet. (Un nombre de host como `www.eurecom.fr`, que termina con el código de país, `.fr`, nos informa de que es probable que el host se encuentre en Francia, pero esto no dice mucho más.) Además, puesto que los nombres de host pueden constar de caracteres alfanuméricos de longitud variable, podrían ser difíciles de procesar por los routers. Por estas razones, los hosts también se identifican mediante **direcciones IP**.

En el Capítulo 4 veremos en detalle las direcciones IP, pero le resultará útil leer ahora una breve exposición sobre las mismas. Una dirección IP consta de cuatro bytes y sigue una rígida estructura jerárquica. El aspecto de una dirección IP es, por ejemplo, `121.7.106.83`, donde cada punto separa uno de los bytes expresados en notación decimal con valores comprendidos entre 0 y 255. Una dirección IP es jerárquica porque al explorar la dirección de izquierda a derecha, se obtiene información cada vez más específica acerca del lugar en el que está ubicado el host dentro de Internet (es decir, en qué red de la red de redes). De forma similar, cuando examinamos una dirección postal de abajo a arriba, obtenemos cada vez información más específica acerca del lugar definido por la dirección.

2.5.1 Servicios proporcionados por DNS

Acabamos de ver que hay dos formas de identificar un host, mediante un nombre de host y mediante una dirección IP. Las personas prefieren utilizar como identificador el nombre de host (más fácil de recordar), mientras que los routers prefieren emplear las direcciones IP de longitud fija y que siguen una estructura jerárquica. Para reconciliar estas preferencias necesitamos un servicio de directorio que traduzca los nombres de host en direcciones IP. Esta es la tarea principal que lleva a cabo el **Sistema de nombres de dominio (DNS, *Domain Name System*)** de Internet. DNS es (1) una base de datos distribuida implementada en una jerarquía de **ser-vidores DNS** y (2) un protocolo de la capa de aplicación que permite a los hosts consultar la base de datos distribuida. Los servidores DNS suelen ser máquinas UNIX que ejecutan el software BIND (*Berkeley Internet Name Domain*, Dominio de nombres de Internet de Berkeley) [BIND 2009]. El protocolo DNS se ejecuta sobre UDP y utiliza el puerto 53.

Otros protocolos de la capa de aplicación, como HTTP, SMTP y FTP, emplean habitualmente DNS para traducir los nombres de host suministrados por el usuario en direcciones IP. Por ejemplo, veamos qué ocurre cuando un navegador (es decir, un cliente HTTP), que se ejecuta en un determinado host de usuario, solicita el URL `www.unaescuela.edu/index.html`. Para que el host del usuario pueda enviar un mensaje de solicitud HTTP al servidor web `www.unaescuela.edu`, el host del usuario debe obtener en primer lugar la dirección IP de `www.unaescuela.edu`. Esto se hace del siguiente modo:

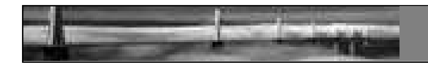
1. La propia máquina cliente ejecuta el lado del cliente de la aplicación DNS.
2. El navegador extrae el nombre de host, www.unaescuela.edu, del URL y pasa el nombre de host al lado del cliente de la aplicación DNS.
3. El cliente DNS envía una consulta que contiene el nombre de host a un servidor DNS.
4. El cliente DNS recibe finalmente una respuesta, que incluye la dirección IP correspondiente al nombre del host.
5. Una vez que el navegador recibe la dirección IP del servidor DNS, puede iniciar una conexión TCP con el proceso servidor HTTP localizado en el puerto 80 en esa dirección.

Podemos ver a partir de este ejemplo que DNS añade un retardo adicional, en ocasiones, sustancial, a las aplicaciones de Internet que le utilizan. Afortunadamente, como veremos más adelante, la dirección IP deseada a menudo está almacenada en caché en un servidor DNS "próximo", lo que ayuda a reducir el tráfico de red DNS, así como el retardo medio del servicio DNS.

DNS proporciona algunos otros servicios importantes además de la traducción de los nombres de host en direcciones IP:

- **Alias de host.** Un host con un nombre complicado puede tener uno o más alias. Por ejemplo, un nombre de host como `relay1.west-coast.enterprise.com` podría tener, digamos, dos alias como `enterprise.com` y `www.enterprise.com`. En este caso, el nombre de host `relay1.west-coast.enterprise.com` se dice que es el **nombre de host canónico**. Los alias de nombres de host, cuando existen, normalmente son más mnemónicos que los nombres canónicos. Una aplicación puede invocar DNS para obtener el nombre de host canónico para un determinado alias, así como la dirección IP del host.
- **Alias del servidor de correo.** Por razones obvias, es enormemente deseable que las direcciones de correo electrónico sean mnemónicas. Por ejemplo, si Benito tiene una cuenta de Hotmail, su dirección de correo puede ser tan simple como `benito@hotmail.com`. Sin embargo, el nombre de host del servidor de correo de Hotmail es más complicado y mucho menos mnemónico que simplemente `hotmail.com` (por ejemplo, el nombre canónico podría ser algo parecido a `relay1.west-coast.hotmail.com`). Una aplicación de correo puede invocar al servicio DNS para obtener el nombre de host canónico para un determinado alias, así como la dirección IP del host. De hecho, el registro MX (véase más adelante) permite al servidor de correo y al servidor web de una empresa tener nombres de host (con alias) iguales; por ejemplo, tanto el servidor web como el servidor de correo de una empresa se pueden llamar `enterprise.com`.
- **Distribución de carga.** DNS también se emplea para realizar la distribución de carga entre servidores replicados, como los servidores web replicados. Los sitios con una gran carga de trabajo, como `cnn.com`, están replicados en varios servidores, ejecutándose cada servidor en un sistema terminal distinto y teniendo cada uno una dirección IP diferente. Para los servidores web replicados hay asociado un *conjunto* de direcciones IP con un único nombre de host canónico. La base de datos DNS contiene este conjunto de direcciones IP. Cuando los clientes realizan una consulta DNS sobre un nombre asignado a un conjunto de direcciones, el servidor responde con el conjunto completo de direcciones IP, pero rota el orden de las direcciones en cada respuesta. Dado que normalmente un cliente envía su mensaje de solicitud HTTP a la dirección IP que aparece en primer lugar dentro del conjunto, la rotación DNS distribuye el tráfico entre los servidores replicados. La rotación DNS también se emplea para el correo electrónico de modo que múltiples servidores de correo pueden tener el mismo alias. Recientemente, las empresas de distribución de contenido como Akamai [Akamai 2009] han utilizado DNS de formas muy sofisticadas para proporcionar la distribución de contenido web (véase el Capítulo 7).

DNS está especificado en los documentos RFC 1034 y RFC 1035, y actualizado en varios RFC adicionales. Se trata de un sistema complejo y aquí sólo hemos tocado los aspectos básicos de su funcionamiento. El lector interesado puede consultar estos RFC y el libro de Abitz y Liu [Abitz 1993]; también puede consultar el documento retrospectivo [Mockapetris 1988], que proporciona una agradable descripción sobre qué es DNS y el por qué de DNS; asimismo puede ver [Mockapetris 2005].



PRÁCTICA

DNS: FUNCIONES CRÍTICAS DE RED MEDIANTE EL PARADIGMA CLIENTE-SERVIDOR

Como HTTP, FTP y SMTP, el protocolo DNS es un protocolo de la capa de aplicación puesto que (1) se ejecuta entre sistemas terminales que están en comunicación utilizando el paradigma cliente-servidor y (2) se basa en un protocolo de transporte subyacente extremo a extremo para transferir los mensajes DNS entre los sistemas terminales en comunicación. Sin embargo, desde otro punto de vista, la función de DNS es bastante diferente a la de las aplicaciones web de transferencia de archivos y de correo electrónico. A diferencia de estas aplicaciones, DNS no es una aplicación con la que el usuario interactúe directamente; en su lugar, DNS lleva a cabo una de las funciones básicas en Internet: traducir los nombres de hosts en sus direcciones IP subyacentes para las aplicaciones de usuario y otras aplicaciones software de Internet. Hemos mencionado en la Sección 1.2 que gran parte de la complejidad de la arquitectura de Internet se encuentra en las "fronteras" de la red. DNS, que implementa el importante proceso de traducción de nombres en direcciones, utilizando clientes y servidores ubicados en la frontera de la red, es otro ejemplo más de dicha filosofía de diseño.

2.5.2 Cómo funciona DNS

Ahora vamos a presentar a alto nivel cómo funciona DNS. Nos centraremos en el servicio de traducción nombre de host-dirección IP.

Suponga que una determinada aplicación (como por ejemplo un navegador web o un lector de correo), que se ejecuta en el host de un usuario, necesita traducir un nombre de host en una dirección IP. La aplicación invocará al lado del cliente de DNS, especificando el nombre de host del que necesita la correspondiente traducción. (En muchos sistemas basados en UNIX, `gethostbyname()` es la llamada a función que una aplicación utiliza para llevar a cabo la traducción. En la Sección 2.7, veremos cómo una aplicación Java puede invocar al servicio DNS.) Entonces, la aplicación DNS en el host del usuario entra en funcionamiento, enviando un mensaje de consulta a la red. Todos los mensajes de consulta y de respuesta DNS se envían dentro de datagramas UDP al puerto 53. Transcurrido un cierto retardo, del orden de milisegundos a segundos, el servicio DNS del host del usuario recibe un mensaje de respuesta DNS que proporciona la traducción deseada, la cual se pasa entonces a la aplicación que lo ha invocado. Por tanto, desde la perspectiva de dicha aplicación que se ejecuta en el host del usuario, DNS es una caja negra que proporciona un servicio de traducción simple y directo. Pero, de hecho, la caja negra que implementa el servicio es compleja, constando de un gran número de servidores DNS distribuidos por todo el mundo, así como de un protocolo de la capa de aplicación que especifica cómo los servidores DNS y los hosts que realizan las consultas se comunican.

Un diseño simple de DNS podría ser un servidor DNS que contuviera todas las correspondencias entre nombres y direcciones. En este diseño centralizado, los clientes simplemente dirigirían todas las consultas a un mismo servidor DNS y éste respondería directamente a las consultas de los clientes. Aunque la simplicidad de este diseño es atractiva, es inapropiado para la red Internet de hoy día a causa de la enorme (y creciente) cantidad de hosts. Entre los problemas con un diseño centralizado podemos citar los siguientes:

- **Un único punto de fallo.** Si el servidor DNS falla, entonces ¡también falla toda la red Internet!
- **Volumen de tráfico.** Un único servidor DNS tendría que gestionar todas las consultas DNS (de todas las solicitudes HTTP y mensajes de correo electrónico generados en cientos de millones de hosts).
- **Base de datos centralizada distante.** Un único servidor DNS no puede "estar cerca" de todos los clientes que efectúan consultas. Si colocamos ese único servidor DNS en la ciudad de Nueva York, entonces todas las consultas desde Australia deberían viajar hasta el otro extremo del globo, quizá a través de enlaces lentos y congestionados. Esto podría llevar por tanto a retardos significativos.
- **Mantenimiento.** Ese único servidor DNS tendría que mantener registros de todos los hosts de Internet. No sólo sería una enorme base de datos centralizada, sino que tendría que ser actualizada con frecuencia con el fin de incluir todos los hosts nuevos.

En resumen, una base de datos centralizada almacenada en un único servidor DNS simplemente *no podría escalarse*. En consecuencia, el sistema DNS utiliza un diseño distribuido. De hecho, DNS es un estupendo ejemplo de cómo se puede implementar una base de datos distribuida en Internet.

Una base de datos jerárquica y distribuida

Para abordar el problema del escalado, DNS utiliza un gran número de servidores, organizados de forma jerárquica y distribuidos alrededor de todo el mundo. Ningún servidor DNS dispone de todas las correspondencias de todos los hosts de Internet. En su lugar, las correspondencias están repartidas por los servidores DNS. En una primera aproximación, podemos decir que existen tres clases de servidores DNS: los servidores DNS raíz, los servidores DNS de dominio de nivel superior (TLD, *Top-Level Domain*) y los servidores DNS autoritativos, organizados en una jerarquía como la mostrada en la Figura 2.19. Con el fin de comprender cómo estas tres clases de servidores interactúan, suponga que un cliente DNS desea determinar la dirección IP correspondiente al nombre de host `www.amazon.com`. En una primera aproximación tienen lugar los siguientes acontecimientos: primero, el cliente contacta con uno de los servidores raíz, el cual devuelve las direcciones IP para los servi-

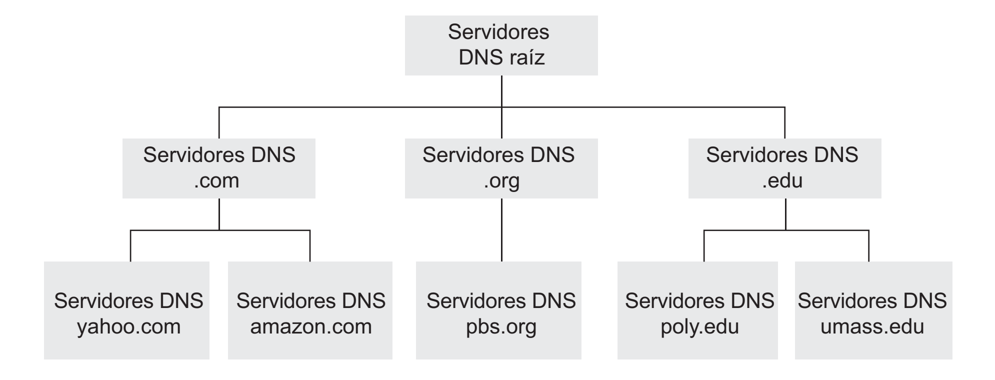


Figura 2.19 • Parte de la jerarquía de los servidores DNS.

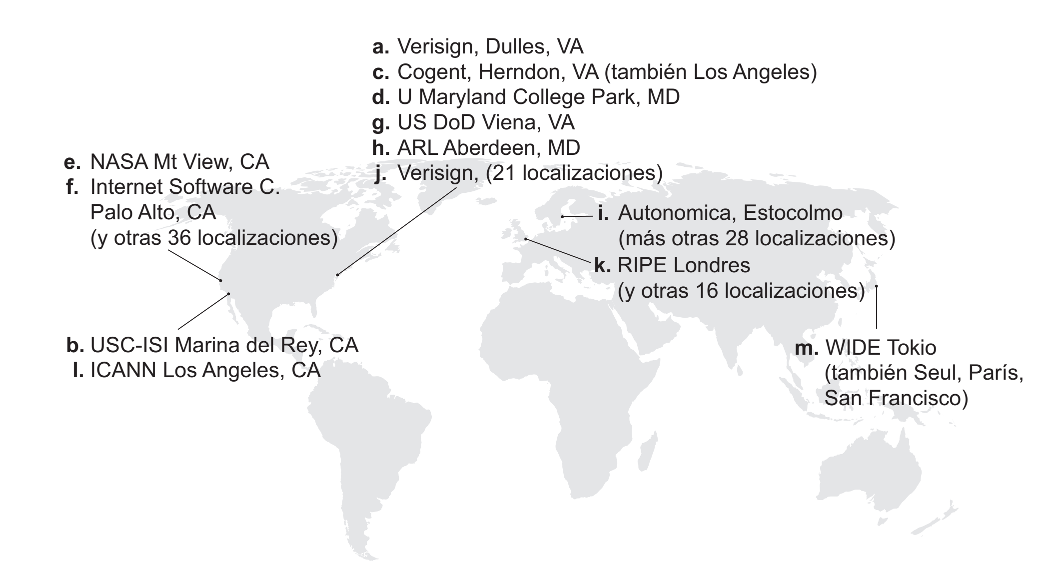


Figura 2.20 • Servidores DNS raíz en 2009 (nombre, organización y ubicación).

dores TLD del dominio de nivel superior `.com`. A continuación, el cliente contacta con uno de estos servidores TLD, que devuelve la dirección IP de un servidor autoritativo para amazon.com. Por último, el cliente contacta con uno de los servidores autoritativos de amazon.com, el cual devuelve la dirección IP correspondiente al nombre de host www.amazon.com. Enseguida examinaremos este proceso de búsqueda DNS con más detalle. Pero en primer lugar, echemos un vistazo a estas tres clases de servidores DNS:

- **Servidores DNS raíz.** En Internet, existen 13 servidores DNS raíz (etiquetados como A hasta M), la mayoría de los cuales se localizan en América del Norte. En la Figura 2.20 se muestra un mapa de octubre de 2006 de los servidores DNS raíz. Hay disponible una lista de los servidores DNS raíz actuales en [Root-servers 2009]. Aunque hemos hecho referencia a cada uno de los 13 servidores DNS raíz como si fueran un único servidor, cada "servidor" es realmente una agrupación (*cluster*) de servidores replicados, tanto por propósitos de seguridad como de fiabilidad.
- **Servidores de dominio de nivel superior (TLD).** Estos servidores son responsables de los dominios de nivel superior, como son com, org, net, edu y gov, y todos los dominios de nivel superior correspondientes a los distintos países, como por ejemplo, uk, fr, ca y jp. La empresa Network Solutions mantiene los servidores TLD para el dominio de nivel superior .com y la empresa Educause mantiene los servidores TLD para el dominio de nivel superior .edu.
- **Servidores DNS autoritativos.** Todas las organizaciones que tienen hosts accesibles públicamente (como son los servidores web y los servidores de correo) a través de Internet deben proporcionar registros DNS accesibles públicamente que establezcan la correspondencia entre los nombres de dichos hosts y sus direcciones IP. Un servidor DNS autoritativo de una organización alberga estos registros DNS. Una organización puede

elegir implementar su propio servidor DNS autoritativo para almacenar estos registros; alternativamente, la organización puede pagar por tener esos registros almacenados en un servidor DNS autoritativo de algún proveedor de servicios. La mayoría de las universidades y de las empresas de gran tamaño implementan y mantienen sus propios servidores DNS autoritativos principal y secundario (*backup*).

Todos los servidores DNS raíz, TLD y autoritativos pertenecen a la jerarquía de servidores DNS, como se muestra en la Figura 2.19. Existe otro tipo importante de servidor DNS conocido como **servidor DNS local**. Un servidor DNS local no pertenece estrictamente a la jerarquía de servidores, pero no obstante es importante dentro de la arquitectura DNS. Cada ISP (como por ejemplo un ISP de una universidad, de un departamento académico, de una empresa o residencial) tiene un servidor DNS local (también denominado servidor de nombres predeterminado). Cuando un host se conecta a un ISP, éste proporciona al host las direcciones IP de uno o más de sus servidores DNS locales (normalmente a través de DHCP, lo que veremos en el Capítulo 4). Usted puede determinar fácilmente la dirección IP de su servidor DNS local accediendo a las ventanas de estado de la red en Windows o UNIX. Generalmente, un servidor DNS local de un host se encuentra "cerca" de ese host. En el caso de un ISP institucional, el servidor DNS local puede encontrarse en la misma LAN que el host; en el caso de un ISP residencial, estará separado del host no más de unos pocos routers. Cuando un host realiza una consulta DNS, ésta se envía al servidor DNS local, que actúa como proxy, reenviando la consulta a la jerarquía de servidores DNS, como veremos más detalladamente a continuación.

Veamos un ejemplo sencillo. Suponga que el host `cis.poly.edu` desea conocer la dirección de `gaia.cs.umass.edu`. Suponga también que el servidor DNS local de la Universidad Politécnica se denomina `dns.poly.edu` y que un servidor DNS autoritativo para `gaia.cs.umass.edu` se llama `dns.umass.edu`. Como se muestra en la Figura 2.21, el host `cis.poly.edu` envía en primer lugar un mensaje de consulta DNS a su servidor DNS local, `dns.poly.edu`. El mensaje de consulta contiene el nombre de host que debe ser traducido, `gaia.cs.umass.edu`. El servidor DNS local reenvía la consulta a un servidor DNS raíz, el cual toma el sufijo `edu` y devuelve al servidor DNS local una lista de las direcciones IP de los servidores TLD responsables del dominio `.edu`. El servidor DNS local reenvía a continuación el mensaje de consulta a uno de estos servidores TLD. El servidor TLD toma nota del sufijo `umass.edu` y responde con la dirección IP del servidor DNS autoritativo correspondiente a la Universidad de Massachusetts, es decir, `dns.umass.edu`. Por último, el servidor DNS local reenvía la consulta directamente a `dns.umass.edu`, que responde con la dirección IP de `gaia.cs.umass.edu`. Observe que en este ejemplo, para obtener la dirección correspondiente a un nombre de host, se han envíado ocho mensajes DNS: ¡cuatro mensajes de consulta y cuatro mensajes de respuesta! Pronto veremos cómo el almacenamiento en caché DNS reduce este tráfico de consultas.

En el ejemplo anterior suponíamos que el servidor TLD conoce el servidor DNS autoritativo correspondiente al nombre de host. En general esto no es así. En lugar de ello, el servidor TLD puede saber únicamente de un servidor DNS intermedio, el cual a su vez sabe cuál es el servidor DNS autoritativo para el nombre de host. Por ejemplo, suponga de nuevo que la Universidad de Massachusetts tiene un servidor DNS para la universidad, denominado `dns.umass.edu`. Suponga también que cada uno de los departamentos de la Universidad de Massachusetts tiene su propio servidor DNS y que cada servidor DNS departamental es un servidor autoritativo para todos los hosts del departamento. En este caso, cuando el servidor DNS intermedio `dns.umass.edu` recibe una consulta para un host

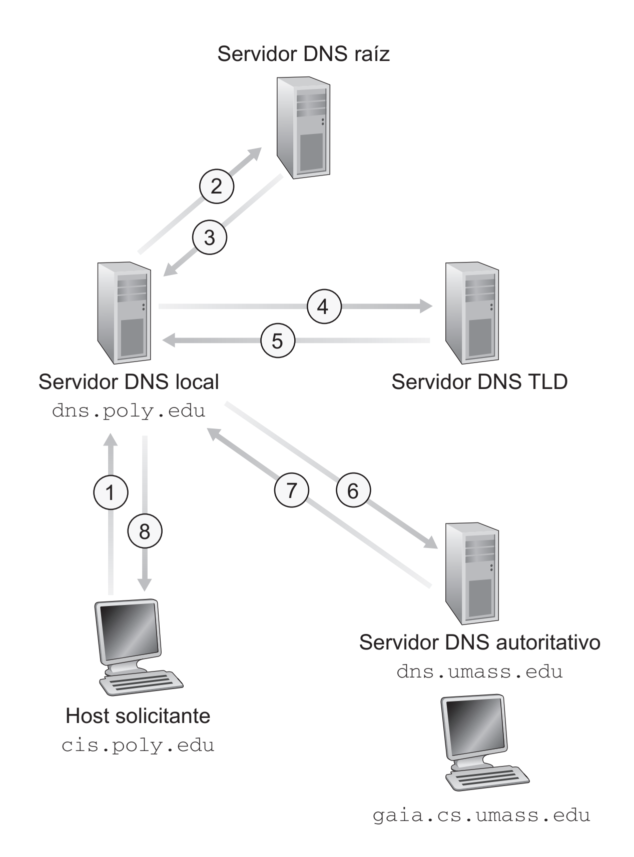


Figura 2.21 • Interacción de los diversos servidores DNS.

cuyo nombre termina en `cs.umass.edu`, devuelve a `dns.poly.edu` la dirección IP de `dns.cs.umass.edu`, que es autoritativo para todos los nombres de host que terminan con `cs.umass.edu`. El servidor DNS local `dns.poly.edu` envía entonces la consulta al servidor DNS autoritativo, que devuelve la correspondencia deseada al servidor DNS local, el cual a su vez devuelve la correspondencia al host que ha hecho la solicitud. En este caso, ¡se han envíado un total de 10 mensajes!

El ejemplo mostrado en la Figura 2.21 utiliza tanto **consultas recursivas** como **consultas iterativas**. La consulta enviada desde `cis.poly.edu` a `dns.poly.edu` es una consulta recursiva, ya que la consulta solicita a `dns.poly.edu` que obtenga por sí mismo la correspondencia. Pero las tres consultas siguientes son iterativas puesto que todas las respuestas son devueltas directamente a `dns.poly.edu`. En teoría, cualquier consulta DNS puede ser iterativa o recursiva. Por ejemplo, la Figura 2.22 muestra una cadena de consultas DNS para las que todas las consultas son recursivas. En la práctica, las consultas normalmente siguen el patrón mostrado en la Figura 2.21: la consulta procedente del host que hace la solicitud al servidor DNS local es recursiva y las restantes consultas son iterativas.

Almacenamiento en caché DNS

Hasta el momento hemos ignorado el **almacenamiento en caché DNS**, una funcionalidad extremadamente importante del sistema DNS. En realidad, DNS explota exhaustivamente el

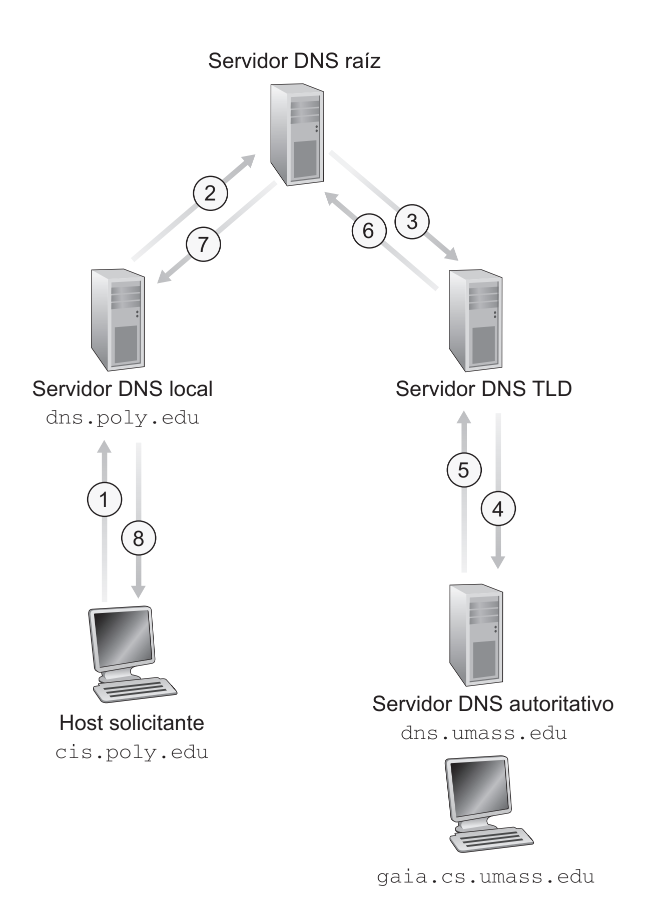


Figura 2.22 • Consultas recursivas en DNS.

almacenamiento en caché para mejorar el comportamiento de los retardos y reducir el número de mensajes DNS que van de un lado a otro por Internet. La idea que se esconde detrás del almacenamiento en caché DNS es muy simple. En una cadena de consultas, cuando un servidor DNS recibe una respuesta DNS (que contiene, por ejemplo, una correspondencia entre un nombre de host y una dirección IP), puede almacenar esta información en su memoria local. Por ejemplo, en la Figura 2.21, cada vez que el servidor DNS local `dns.poly.edu` recibe una respuesta de algún servidor DNS, puede almacenar en caché cualquier información contenida en esa respuesta. Si una relación nombre de host/dirección IP se almacena en la caché de un servidor DNS y llegan otras consultas a ese mismo servidor DNS preguntando por el mismo nombre de host, el servidor DNS puede proporcionar la dirección IP deseada, incluso aunque no sea autoritativo para el nombre de host. Dado que los hosts y las correspondencias entre nombres de host y direcciones IP no son permanentes, los servidores DNS descartan la información almacenada en caché pasado un cierto periodo de tiempo (normalmente, unos dos días).

Por ejemplo, suponga que un host `apricot.poly.edu` consulta a `dns.poly.edu` solicitándole la dirección IP correspondiente al nombre de host `cnn.com`. Suponga también que unas pocas horas más tarde, otro host de la Universidad Politécnica, digamos

kiwi.poly.fr, también hace una consulta a dns.poly.edu especificando el mismo nombre de host. Gracias al almacenamiento en caché, el servidor DNS local podrá devolver de forma inmediata la dirección IP de cnn.com al segundo host que la ha solicitado sin tener que consultar a ningún otro servidor DNS. Un servidor DNS local también puede almacenar en caché las direcciones IP de los servidores TLD, permitiendo así a los servidores DNS locales saltarse a los servidores DNS raíz en una cadena de consultas (lo que ocurre con frecuencia).

2.5.3 Registros y mensajes DNS

Los servidores DNS que implementan conjuntamente la base de datos distribuida DNS almacenan los **registros de recursos (RR)**, incluyendo los que proporcionan las correspondencias entre nombre de host y dirección IP. Cada mensaje de respuesta DNS transporta uno o más registros de recursos. En ésta y en la subsección siguiente, proporcionamos una breve introducción a los recursos y mensajes DNS; puede encontrar información detallada en [Abitz 1993] o en los RFC sobre DNS [RFC 1034; RFC 1035].

Un registro de recurso está formado por los siguientes cuatro campos:

```
(Nombre, Valor, Tipo, TTL)
```

TTL es el tiempo de vida del registro de recurso; determina cuándo un recurso debería ser eliminado de una caché. En los registros de ejemplo dados a continuación, hemos ignorado el campo TTL. El significado de Nombre y Valor depende del campo Tipo:

- Si Tipo=A, entonces Nombre es un nombre de host y Valor es la dirección IP correspondiente a dicho nombre. Por tanto, un registro de tipo A proporciona la correspondencia estándar nombre de host-dirección IP. Por ejemplo, (relay1.bar.foo.com, 145.37.93.126, A) es un registro de tipo A.
- Si Tipo=NS, entonces Nombre es un dominio (como foo.com) y Valor es el nombre de host de un servidor DNS autoritativo que sabe cómo obtener las direcciones IP de los hosts del dominio. Este registro se utiliza para encaminar las consultas DNS a lo largo de la cadena de consultas. Por ejemplo, (foo.com, dns.foo.com, NS) es un registro de tipo NS.
- Si Tipo=CNAME, entonces Valor es un nombre de host canónico correspondiente al alias especificado por Nombre. Este registro puede proporcionar a los hosts que hacen consultas el nombre canónico correspondiente a un nombre de host. Por ejemplo, (foo.com, relay1.bar.foo.com, CNAME) es un registro CNAME.
- Si Tipo=MX, entonces Valor es el nombre canónico de un servidor de correo que tiene un alias dado por Nombre. Por ejemplo, (foo.com. mail.bar.foo.com, MX) es un registro MX. Los registros MX permiten a los nombres de host de los servidores de correo tener alias simples. Observe que utilizando el registro MX, una empresa puede tener el mismo alias para su servidor de correo y para uno de sus otros servidores (como por ejemplo, el servidor web). Para obtener el nombre canónico del servidor de correo, un cliente DNS consultaría un registro MX y para conocer el nombre canónico del otro servidor, consultaría el registro CNAME.

Si un servidor DNS es autoritativo para un determinado nombre de host, entonces el servidor DNS contendrá un registro de tipo A para el nombre de host. (Incluso aunque

el servidor DNS no sea autoritativo, puede contener un registro de tipo A en su caché.) Si un servidor no es autoritativo para un nombre de host, entonces el servidor contendrá un registro de tipo NS para el dominio que incluye el nombre de host; también contendrá un registro de tipo A que proporcione la dirección IP del servidor DNS en el campo `Valor` del registro NS. Por ejemplo, suponga un servidor TLD `edu` que no es autoritativo para el host `gaia.cs.umass.edu`. Por tanto, este servidor contendrá un registro para un dominio que incluye el host `gaia.cs.umass.edu`, por ejemplo, `(umass.edu, dns.umass.edu, NS)`. El servidor TLD `edu` también contendría un registro de tipo A, que establece la correspondencia entre el servidor DNS `dns.umass.edu` y una dirección IP, como en `(dns.umass.edu, 128.119.40.111, A)`.

Mensajes DNS

Anteriormente en esta sección hemos hecho referencia a los mensajes de respuesta y consultas DNS. Únicamente existen estas dos clases de mensajes DNS. Además, tanto los mensajes de respuesta como de consulta utilizan el mismo formato, que se muestra en la Figura 2.23. La semántica en los distintos campos de un mensaje DNS es la siguiente:

- Los primeros 12 bytes constituyen la *sección de cabecera,* la cual contiene una serie de campos. El primero de estos campos es un número de 16 bits que identifica la consulta. Este identificador se copia en el mensaje de respuesta a la consulta, lo que permite al cliente establecer las correspondencias correctas entre las respuestas recibidas y las consultas enviadas. En el campo Indicadores se incluyen, como su nombre indica, una serie de indicadores. Un indicador consulta/respuesta de 1 bit informa de si el mensaje es una consulta (0) o una respuesta (1). Un indicador autoritativo de 1 bit se activa en un mensaje de respuesta cuando un servidor DNS es un servidor autoritativo para un nombre soli-citado.

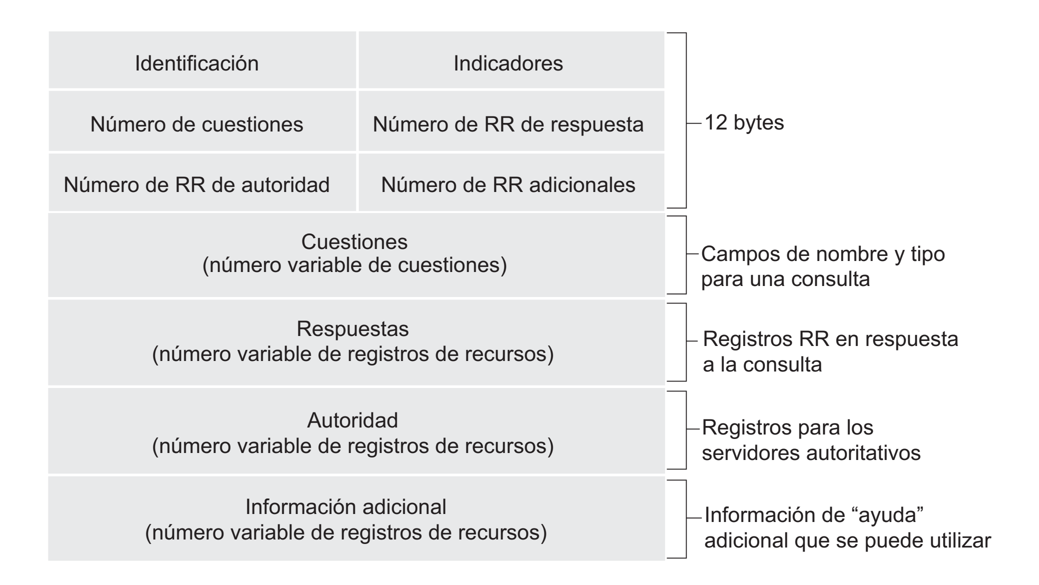


Figura 2.23 • Formato de los mensajes DNS.

El indicador recursión-deseada, también de 1 bit, se activa cuando un cliente (host o servidor DNS) desea que el servidor DNS realice una recursión cuando no disponga del registro. En un mensaje de respuesta, el campo de recursión-disponible de 1 bit se activa si el servidor DNS soporta la recursión. En la cabecera también se incluyen cuatro campos "número de", que indican el número de apariciones de los cuatro tipos de secciones de datos que siguen a la cabecera.

- La *sección cuestiones* contiene información acerca de la consulta que se va a realizar. Esta sección incluye (1) un campo de nombre que contiene el nombre que se va a consultar y (2) un campo de tipo que especifica el tipo de cuestión que se plantea acerca del nombre; por ejemplo, la dirección del host asociada con un nombre (tipo A) o el servidor de correo para un nombre (tipo MX).
- En una respuesta de un servidor DNS, la *sección respuestas* contiene los registros del recurso para el nombre que fue consultado originalmente. Recuerde que en cada registro de recurso existe un parámetro `Tipo` (por ejemplo, A, NS, CNAME y MX), un parámetro `Valor` y el parámetro `TTL`. Una respuesta puede devolver varios registros de recursos, ya que un nombre de host puede tener asociadas varias direcciones IP (por ejemplo, para los servidores web replicados, como hemos visto anteriormente en esta sección).
- La *sección autoridad* contiene registros de otros servidores autoritativos.
- La sección *información adicional* contiene otros registros útiles. Por ejemplo, el campo de respuesta en un mensaje de respuesta a una consulta MX contiene un registro de recurso que proporciona el nombre de host canónico de un servidor de correo. Esta sección de información adicional contiene un registro de tipo A que proporciona la dirección IP para el nombre de host canónico del servidor de correo.

¿Le gustaría enviar un mensaje de consulta DNS directamente desde el host en el que está trabajando a algún servidor DNS? Esto podría hacerse fácilmente con el **programa nslookup**, que está disponible en la mayoría de las plataformas Windows y UNIX. Por ejemplo, en un host Windows basta con abrir la aplicación Símbolo del sistema (*prompt*) e invocar al programa nslookup escribiendo simplemente "nslookup". A continuación, podrá enviar una consulta DNS a cualquier servidor DNS (raíz, TLD o autoritativo). Después de recibir el mensaje de respuesta del servidor DNS, nslookup mostrará los registros incluidos en la respuesta (en un formato legible para las personas). Como alternativa a la ejecución de nslookup desde su propio host, puede visitar uno de los muchos sitios web que permiten utilizar nslookup de forma remota (basta con escribir "nslookup" en un motor de búsqueda para acceder a uno de estos sitios).

Inserción de registros en la base de datos DNS

La exposición anterior se ha centrado en cómo se recuperan los registros de la base de datos DNS. Es posible que ahora se esté preguntando cómo se introducen los registros en dicha base de datos. Veamos cómo se hace esto en el contexto de un ejemplo concreto. Suponga que hemos creado una nueva empresa llamada Network Utopia. Lo primero que seguramente deseará hacer es registrar el nombre de dominio `networkutopia.com` en un registro. Un **registrador** es una entidad comercial que verifica la unicidad del nombre de dominio, lo añade a la base de datos DNS (como veremos más adelante) y percibe unas pequeñas tasas por sus servicios. Antes de 1999, un único registrador, Network Solutions,

tenía el monopolio sobre el registro de nombres para los dominios com, net y org. Pero actualmente, existen muchas entidades registradoras que compiten por los clientes. La ICANN (*Internet Corporation for Assigned Names and Numbers*, Corporación de Internet para nombres y números asignados) acredita a las distintas entidades registradoras. En la dirección http://www.internic.net. puede encontrar una lista completa de estas entidades.

Al registrar el nombre de dominio networkutopia.com en alguna entidad registradora, también tendrá que proporcionarle los nombres y direcciones IP de sus servidores DNS autoritativos principal y secundario. Suponga que en nuestro ejemplo estos datos son: dns1.networkutopia.com, dns2.networkutopia.com, 212.212.212.1 y 212.212.212.2. Para cada uno de estos dos servidores el registrador se asegura de que se introduzca un registro de tipo NS y un registro de tipo A en los servidores TLD com. Específicamente, para el servidor autoritativo principal de networkutopia.com, la entidad registradora deberá insertar los siguientes dos registros de recursos en el sistema DNS:

```
(networkutopia.com, dns1.networkutopia.com, NS)

(dns1.networkutopia.com, 212.212.212.1, A)
```

También tendrá que asegurarse de que el registro de recurso de tipo A para su servidor web www.networkutopia.com y el registro de recurso de tipo MX para su servidor de correo mail.networkutopia.com se han introducido en sus servidores DNS autoritativos. (Hasta hace poco, los contenidos de los servidores DNS se configuraban estáticamente; por ejemplo, a partir de un archivo de configuración creado por un administrador del sistema. Sin embargo, recientemente se ha añadido una opción de actualización (UPDATE) al protocolo DNS que permite añadir o borrar dinámicamente los datos de la base de datos mediante mensajes DNS. En los documentos [RFC 2136] y [RFC 3007] se especifican las actualizaciones dinámicas de DNS.)

Una vez que haya completado estos pasos, los usuarios podrán visitar su sitio web y enviar correos electrónicos a los empleados de su empresa. Vamos a terminar esta exposición verificando que esta afirmación es cierta. Esta verificación también le ayudará a consolidar lo que ha aprendido sobre DNS. Suponga que Alicia se encuentra en Australia y desea ver la página web www.networkutopia.com. Como hemos explicado anteriormente, su host enviará en primer lugar una consulta DNS a su servidor DNS local, el cual a su vez se pondrá en contacto con un servidor TLD com. (El servidor DNS local también tendrá que comunicarse con un servidor DNS raíz si no tiene en caché la dirección de un servidor TLD com.) El servidor TLD contendrá los registros de recursos de tipo NS y de tipo A enumerados anteriormente, ya que la entidad registradora los habrá almacenado en todos los servidores TLD com. El servidor TLD com envía entonces una respuesta al servidor DNS local de Alicia, conteniendo dicha respuesta los dos registros de recursos. A continuación, el servidor DNS local transmite una consulta DNS a 212.212.212.1, solicitando el registro de tipo A correspondiente a www.networkutopia.com. Este registro proporciona la dirección IP del servidor web deseado, por ejemplo, 212.212.71.4, que el servidor DNS local pasa al host de Alicia. En este momento, el navegador de Alicia puede iniciar una conexión TCP con el host 212.212.71.4 y enviar una solicitud HTTP a través de la misma. Como ha podido ver, son muchas las cosas que suceden entre bastidores cuando navegamos por la Web.



SEGURIDAD

VULNERABILIDADES DNS

Hemos visto que DNS es un componente fundamental de la infraestructura de Internet y que muchos servicios importantes, entre los que incluyen las aplicaciones web y de correo electrónico no podrían funcionar sin él. Por tanto, lo natural es preguntarse: ¿Cómo puede ser atacado DNS? ¿Es DNS un blanco fácil, que espera a ser puesto fuera de servicio, arrastrando con él a la mayoría de las aplicaciones de Internet?

El primer tipo de ataque que nos viene a la mente es un ataque DDoS por inundación de ancho de banda (véase la Sección 1.6) contra los servidores DNS. Por ejemplo, un atacante podría intentar enviar a cada uno de los servidores DNS raíz una gran cantidad de paquetes, tantos que la mayor parte de las consultas DNS legítimas nunca fueran contestadas. Un ataque DDoS a gran escala contra servidores DNS raíz tuvo lugar realmente el 21 de octubre de 2002. En ese ataque, los atacantes utilizaron una botnet para enviar enormes cargas de mensajes ping ICMP a los 13 servidores DNS raíz. (Los mensajes ICMP se estudian en el Capítulo 4. Por el momento, nos basta con saber que los paquetes ICMP son tipos especiales de datagramas IP.) Afortunadamente, este ataque a gran escala causó unos daños mínimos, sin tener apenas impacto sobre la experiencia en Internet de los usuarios. Los atacantes dirigieron con éxito gran cantidad de paquetes a los servidores raíz, pero muchos de estos servidores estaban protegidos mediante mecanismos de filtrado de paquetes configurados para bloquear siempre todos los mensajes ping ICMP dirigidos a los mismos. Estos servidores protegidos estaban resguardados y funcionaron normalmente. Además, la mayoría de los servidores DNS locales tenían almacenadas en caché las direcciones IP de los servidores de dominio de nivel superior, permitiendo el procesamiento de consultas ignorando normalmente a los servidores DNS raíz.

Un ataque DDoS potencialmente más efectivo contra servidores DNS consistiría en enviar una gran cantidad de consultas DNS a los servidores de dominio de alto nivel; por ejemplo, a todos aquellos servidores que administren el dominio .com. Sería bastante complicado filtrar las consultas DNS dirigidas a servidores DNS; y los servidores de dominio de nivel superior no pueden ser puenteados tan fácilmente como los servidores raíz. Pero la severidad de un ataque así podría ser parcialmente mitigada por el almacenamiento en caché de los servidores DNS locales.

El sistema DNS también podría ser atacado en teoría de otras maneras. En un ataque por intermediación (*man-in-the-middle*), el atacante intercepta las consultas de los hosts y devuelve respuestas falsas. En el ataque por envenenamiento DNS, el atacante envía respuestas falsas a un servidor DNS, engañándole y haciendo que almacene en su caché registros falsos. Cualquiera de estos ataques podría utilizarse, por ejemplo, para redirigir a un usuario web inadvertido al sitio web del atacante. Sin embargo, estos ataques son difíciles de implementar, ya que requieren interceptar los paquetes o engañar a los servidores [Skoudis 2006].

Otro importante ataque DNS no es un ataque al servicio DNS en sí, sino que se aprovecha de la infraestructura de DNS para lanzar un ataque DDoS contra un host objetivo (por ejemplo, el servidor de correo de una universidad). En este ataque, el atacante envía consultas DNS a muchos servidores DNS autoritativos, incluyendo en cada consulta como dirección de origen la del host objetivo. Los servidores DNS enviarán entonces sus respuestas directamente al host objetivo. Si las consultas pueden crearse de tal forma que una respuesta sea mucho más larga (en bytes) que una consulta (lo que se conoce como amplificación), entonces el atacante



SEGURIDAD

podría potencialmente colapsar al host objetivo sin tener que generar él mismo gran parte de ese tráfico. Hasta la fecha, estos ataques por reflexión que explotan DNS han tenido un éxito limitado [Mirkovic 2005].

En resumen, DNS ha demostrado ser sorprendentemente robusto frente a los ataques. Hasta el momento, no ha habido ningún ataque que haya conseguido interrumpir con éxito el servicio DNS. Por el contrario, sí han tenido éxito ataques por reflexión; no obstante, este tipo de ataques pueden ser (y están siendo) controlados mediante la apropiada configuración de los servidores DNS.



2.6 Aplicaciones P2P

Las aplicaciones descritas en este capítulo hasta el momento, incluyendo las aplicaciones web, de correo electrónico y DNS, emplean todas ellas arquitecturas cliente-servidor que dependen en gran medida de que exista una infraestructura de servidores siempre activos. Recuerde de la Sección 2.1.1 que con una arquitectura P2P no se depende más que mínimamente (o no se depende en absoluto) de que exista esa infraestructura de servidores siempre activos. En su lugar, una serie de parejas de hosts conectados de forma intermitente, denominados pares o *peers*, se comunican directamente entre sí. Los pares no son propiedad de un proveedor de servicios, sino que son computadoras de escritorio o portátiles controlados por los usuarios.

En esta sección examinaremos tres tipos de aplicaciones diferentes que están particularmente bien adaptadas a los diseños P2P. La primera de ellas es la distribución de archivos, donde la aplicación distribuye un archivo desde un único origen a un gran números de pares. La distribución de archivos es un buen punto de partida para investigar el funcionamiento de P2P, ya que muestra de forma clara la característica de auto-escalabilidad de las arquitecturas P2P. Como ejemplo específico de la distribución de archivos, vamos a describir el popular sistema BitTorrent. Con la segunda aplicación P2P, examinaremos una base de datos distribuida a lo largo de una comunidad de peers de gran tamaño. Para esta aplicación, usaremos el concepto de Tabla hash distribuida (DHT, *Distributed Hash Table*). Por último y como tercera aplicación, estudiaremos Skype, una aplicación de telefonía por Internet P2P de gran éxito.

2.6.1 Distribución de archivos P2P

Comenzaremos nuestra incursión en P2P considerando una aplicación muy corriente, la distribución de un archivo de gran tamaño desde un único servidor a muchos otros hosts. El archivo podría ser una nueva versión del sistema operativo Linux, un parche software para una aplicación o un sistema operativo existentes, un archivo de audio MP3 o un archivo de vídeo MPEG. En la distribución de archivos cliente-servidor, el servidor debe enviar una copia del archivo a cada uno de los pares, provocando una enorme sobrecarga en el servidor y consumiendo una gran cantidad de su ancho de banda. En la distribución de archivos P2P, cada par puede redistribuir cualquier parte del archivo que ha recibido a cualesquiera otros pares, ayudando de este modo al servidor a llevar a cabo el proceso de distribución

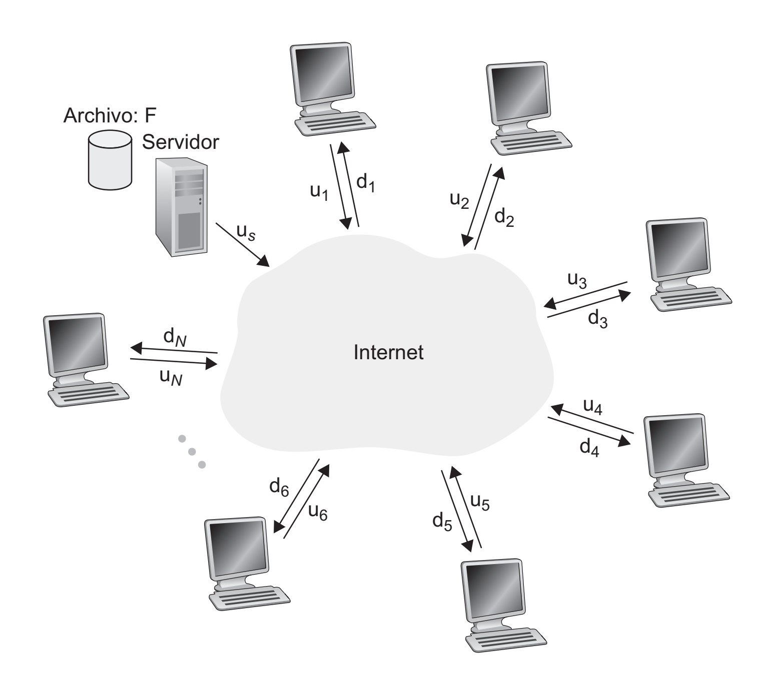


Figura 2.24 • Problema de distribución de un archivo.

En el momento de escribir estas líneas (otoño de 2009), el protocolo de distribución de archivos P2P más popular es BitTorrent [BitTorrent 2009]. Originalmente, fue desarrollado por Bram Cohen (véase la entrevista a Bram Cohen al final del capítulo), pero ahora existen muchos clientes BitTorrent independientes distintos que cumplen con el protocolo BitTorrent, al igual que existen diversos navegadores web que cumplen el protocolo HTTP. En esta subsección examinaremos en primer lugar la característica de auto-escalabilidad de las arquitecturas P2P en el contexto de la distribución de archivos. Después describiremos en detalle BitTorrent, destacando sus funcionalidades y características más importantes.

Escalabilidad de las arquitecturas P2P

Con el fin de comparar las arquitecturas cliente-servidor con las arquitecturas P2P y para ilustrar la auto-escalabilidad inherente de P2P, ahora vamos a considerar un modelo cuantitativo simple para la distribución de un archivo a un conjunto fijo de pares en ambos tipos de arquitectura. Como se muestra en la Figura 2.24, el servidor y los pares están conectados a Internet mediante enlaces de acceso. Sea u_s la velocidad de carga del enlace de acceso del servidor, u_i la velocidad de carga del enlace de acceso del par i y d_i la velocidad de descarga del enlace de acceso del par i. Sea F el tamaño en bits del archivo que se va a distribuir y N el número de pares que desean obtener una copia del archivo. El **tiempo de distribución** es el tiempo que tardan los N pares en obtener una copia del archivo. En el análisis del tiempo de distribución que proporcionamos a continuación, para ambas arquitecturas, hemos hecho una simplificación (pero generalmente precisa [Akella 2003]): suponer que el núcleo de Internet tiene el ancho de banda suficiente, lo que implica que todos los cuellos de botella se encuentran en el acceso a red. También hemos supuesto que el servidor y los clientes no están participando en ninguna otra aplicación de red, de modo que

los anchos de banda para carga y descarga están dedicados completamente a distribuir el archivo.

En primer lugar, vamos a determinar el tiempo de distribución para la arquitectura cliente-servidor, el cual denotaremos como D_{cs}. En esta arquitectura, ninguno de los pares ayudan a distribuir el archivo. Tenemos que hacer las dos observaciones siguientes:

- El servidor debe transmitir una copia del archivo a cada uno de los N pares. Por tanto, el servidor tiene que transmitir NF bits. Puesto que la velocidad de carga del servidor es u_s, el tiempo para distribuir el archivo tiene que ser como mínimo NF/u_s.
- Sea $d_{\text{mín}}$ la velocidad de descarga del par cuya velocidad de descarga sea menor; es decir, $d_{\text{mín}} = \text{mín}\{d_1,d_p,\ldots,d_N\}$. El par con la menor velocidad de descarga no puede obtener los F bits del archivo en menos de $F/d_{\text{mín}}$ segundos. Por tanto, el tiempo mínimo de distribución es, al menos igual a $F/d_{\text{mín}}$.

Teniendo en cuenta estas dos observaciones, se obtiene:

$$D_{cs} \quad \text{máx} \left\{ \frac{NF}{u_s}, \frac{F}{d_{\text{mín}}} \right\}.$$

Esto proporciona un límite inferior al tiempo de distribución para la arquitectura cliente-servidor. En los problemas de repaso se le pedirá que demuestre que el servidor puede planificar sus transmisiones de manera que el límite inferior sea alcanzado realmente. Por tanto, tomemos este límite inferior como el tiempo de distribución real, es decir,

$$D_{cs} = \text{máx} \left\{ \frac{NF}{u_s}, \frac{F}{d_{\text{mín}}} \right\} \tag{2.1}$$

A partir de la Ecuación 2.1, se ve que para N lo suficientemente grande, el tiempo de distribución en una arquitectura cliente-servidor está dada por NF/u_s. Por tanto, el tiempo de distribución aumenta linealmente con el número de pares N. Así, por ejemplo, si el número de pares se multiplica por mil en una semana, pasando de mil a un millón, el tiempo necesario para distribuir el archivo a todos los pares se verá multiplicado por 1.000.

Hagamos ahora un análisis similar para la arquitectura P2P, donde cada par puede ayudar al servidor a distribuir el archivo. Cuando un par recibe datos del archivo, puede utilizar su propia capacidad de carga para redistribuir los datos a otros pares. Calcular el tiempo de distribución para la arquitectura P2P es algo más complicado que para la arquitectura cliente-servidor, ya que el tiempo de distribución depende de cómo cada par implicado distribuya partes del archivo a los demás pares. No obstante, puede obtenerse una expresión simple que permite calcular el tiempo mínimo de distribución [Kumar 2006]. Para este fin, debemos tener en cuenta las siguientes observaciones:

- Al comenzar el proceso de distribución, el archivo sólo lo tiene el servidor. Para que este archivo llegue a la comunidad de pares, el servidor tiene que enviar cada bit del archivo al menos una vez por su enlace de acceso. Por tanto, el tiempo mínimo de distribución es, como mínimo, F/u_s. (A diferencia de lo que ocurre en el esquema cliente-servidor, un bit enviado por el servidor puede no tener que ser enviado de nuevo por el mismo, ya que los pares pueden distribuirlo entre ellos.)

- Al igual que en la arquitectura cliente-servidor, el par con la menor velocidad de descarga no puede obtener los F bits del archivo en menos de $F/d_{\text{mín}}$ segundos. Por tanto, el tiempo mínimo de distribución es al menos igual a $F/d_{\text{mín}}$.
- Por último, observe que la capacidad total de carga del sistema como un todo es igual a la velocidad de carga del servidor más las velocidades de carga de cada par, es decir, $u_{\text{total}} = u_s + u_1 + \ldots + u_N$. El sistema tiene que suministrar (cargar) F bits en cada uno de los N peers, suministrando en total NF bits. Esto no se puede hacer a una velocidad mayor que u_{total}; por tanto, el tiempo mínimo de distribución es también mayor o igual que $NF/(u_s + u_1 + \ldots + u_N)$.

Teniendo en cuenta estas tres observaciones, obtenemos el tiempo mínimo de distribución para la arquitectura P2P, D_{P2P}.

$$D_{\text{P2P}} \geq \text{máx} \left\{ \frac{F}{u_s}, \frac{F}{d_{\text{mín}}}, \frac{NF}{u_s + \sum_{i=1}^{N} u_i} \right\} \tag{2.2}$$

La Ecuación 2.2 proporciona un límite inferior para el tiempo mínimo de distribución en una arquitectura P2P. Si suponemos que cada peer puede redistribuir un bit tan pronto como lo recibe, entonces existe un esquema de redistribución que permite alcanzar este límite inferior [Kumar 2006]. (Demostraremos un caso especial de este resultado en los problemas de repaso.) En realidad, cuando se redistribuyen fragmentos del archivo en lugar de bits individuales, la Ecuación 2.2 sirve como una buena aproximación del tiempo mínimo de distribución real. Por tanto, vamos a tomar el límite inferior dado por la Ecuación 2.2 como el tiempo mínimo de distribución real, es decir,

$$D_{\text{P2P}} = \text{máx} \left\{ \frac{F}{u_s}, \frac{F}{d_{\text{mín}}}, \frac{NF}{u_s + \sum_{i=1}^{N} u_i} \right\} \tag{2.3}$$

La Figura 2.25 compara el tiempo mínimo de distribución de las arquitecturas cliente-servidor y P2P, suponiendo que todos los pares tienen la misma velocidad de carga u. En la figura, hemos establecido que $F/u = 1$ hora, $u_s = 10u$ y $d_{\text{mín}}$ u_s. Por tanto, un par puede transmitir el archivo completo en una hora, la velocidad de transmisión del servidor es 10 veces la velocidad de carga del par y (para simplificar) las velocidades de descarga de los pares son lo suficientemente grandes como para no tener ningún efecto. A partir de la Figura 2.25, podemos ver que para la arquitectura cliente-servidor el tiempo de distribución aumenta linealmente y sin límite a medida que el número de pares aumenta. Sin embargo, en una arquitectura P2P, el tiempo mínimo de distribución no sólo siempre es menor que el tiempo de distribución en la arquitectura cliente-servidor; también es menor que una hora para *cualquier* número N de pares. Por tanto, las aplicaciones que emplean arquitectura P2P pueden auto-escalarse. Esta escalabilidad es una consecuencia directa de que los pares actúan a la vez como redistribuidores y consumidores de bits.

BitTorrent

BitTorrent es un popular protocolo P2P para la distribución de archivos [BitTorrent 2009]. En la jerga de BitTorrent, la colección de todos los pares que participan en la distri-

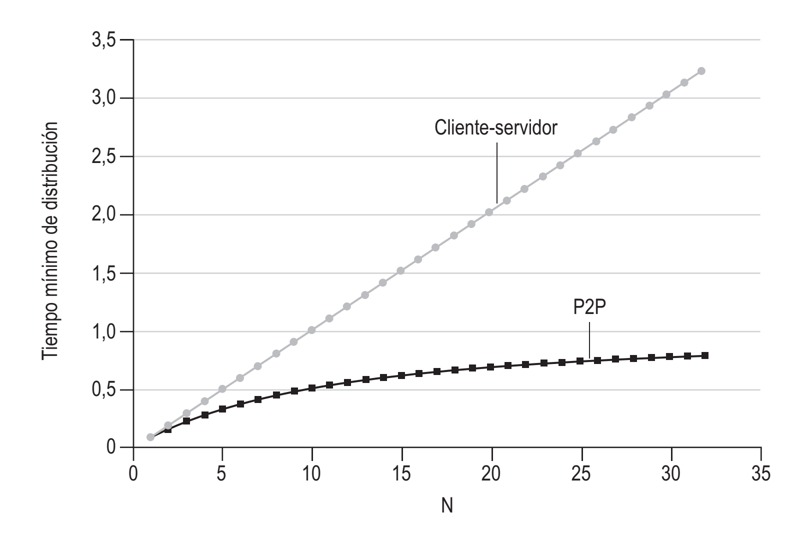


Figura 2.25 • Tiempo de distribución para las arquitecturas P2P y cliente-servidor.

bución de un determinado archivo se conoce como *torrent* (torrente). Los peers de un torrente descargan *fragmentos* del mismo tamaño del archivo de uno a otro, siendo el tamaño típico de un fragmento de 256 kBytes. Cuando un par se une por primera vez a un torrente, no tiene fragmentos del archivo. A lo largo del tiempo va acumulando cada vez más fragmentos. A la vez que descarga fragmentos, actualiza fragmentos en otros pares. Una vez que un par ha adquirido el archivo completo, puede (egoístamente) abandonar el torrente, o (de forma altruista) permanecer en el mismo y continuar suministrando fragmentos a otros pares. Además, cualquier par puede abandonar el torrente en cualquier instante con sólo un subconjunto de fragmentos, y volver a unirse más tarde.

Veamos ahora más de cerca cómo opera BitTorrent. Puesto que BitTorrent es un sistema y protocolo bastante complejo, sólo vamos a describir sus mecanismos más importantes, vamos a dejar al margen algunos detalles con el fin de poder ver claramente cómo funciona. Cada torrente tiene un nodo de infraestructura denominado *tracker* (rastreador). Cuando un par se une a un torrente, se registra mediante el tracker y, periódicamente, informa de que todavía se encuentra en el torrente. De esta manera, el tracker sigue la pista a los pares que están participando en el torrente. Un determinado torrente puede tener en un instante dado un número de pares participantes tan bajo como diez o tan alto como mil.

Como se muestra en la Figura 2.26, cuando un nuevo par, Alicia, se une al torrente, el tracker selecciona aleatoriamente un subconjunto de pares (digamos por ejemplo 50, con el fin de concretar) del conjunto de peers participantes y envía las direcciones IP de estos 50 peers a Alicia. Teniendo en su poder esta lista de pares, Alicia intenta establecer conexiones TCP concurrentes con todos los pares incluidos en dicha lista. Denominaremos a todos los pares con los que Alicia consigue establecer con éxito una conexión TCP “pares vecinos”. (En la Figura 2.26 vemos que Alicia sólo tiene tres pares vecinos. Normalmente, podría tener muchos más.) A medida que pasa el tiempo, algunos de estos pares pueden abandonar la conexión y otros (aparte de los 50 iniciales) pueden intentar establecer conexiones TCP con Alicia. Por tanto, los pares vecinos de un determinado par irán variando con el tiempo.

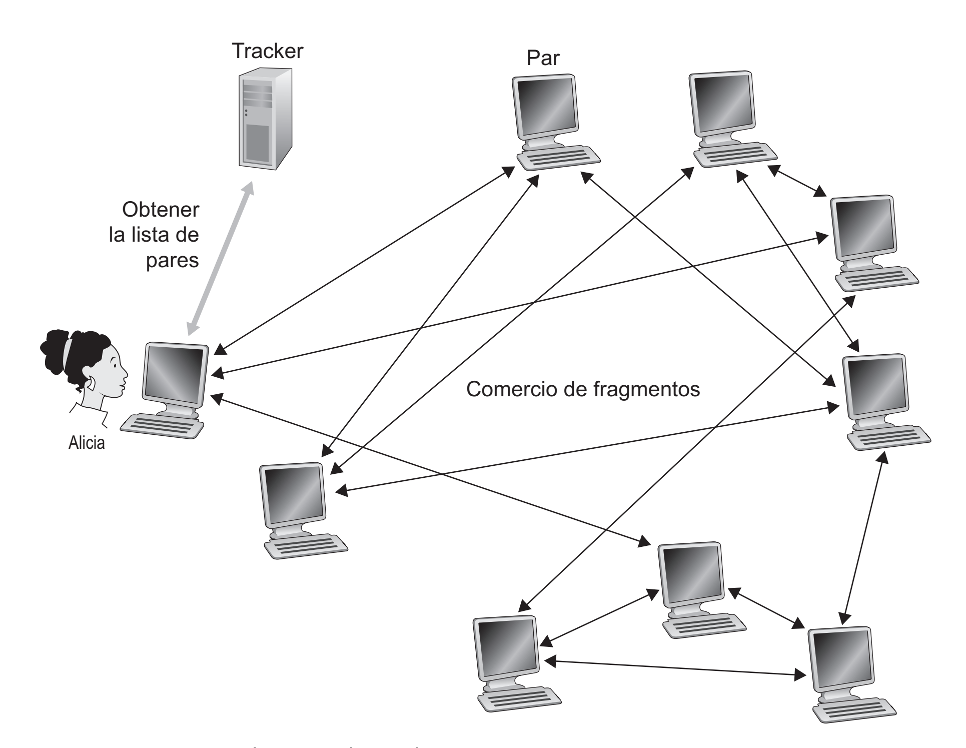


Figura 2.26 • Distribución de archivos con BitTorrent.

En un determinado instante de tiempo, cada peer tendrá un subconjunto de fragmentos del archivo, disponiendo los distintos pares de subconjuntos diferentes. Periódicamente, Alicia preguntará a cada uno de sus vecinos (a través de las conexiones TCP) por la lista de fragmentos de la que disponen. Si Alicia tiene *L* vecinos diferentes obtendrá *L* listas de fragmentos. Con esta información, Alicia solicitará (a través de las conexiones TCP, claro está) los fragmentos que ella no tiene.

De esta manera, en un determinado instante, Alicia tendrá un subconjunto de fragmentos y sabrá qué fragmentos tienen sus vecinos. Con esta información, Alicia tendrá que tomar dos importantes decisiones. En primer lugar, qué fragmentos debe solicitar primero a sus vecinos. Y en segundo lugar, a cuáles de sus vecinos debe enviar ella los fragmentos solicitados. Para decidir qué fragmentos solicitar, Alicia utiliza una técnica conocida como **primero el menos común**. La idea es determinar, de entre los fragmentos que ella no tiene, los fragmentos menos comunes entre sus vecinos (es decir, los fragmentos de los que existe el menor número de copias repetidas repartidas entre los vecinos) y solicitar entonces en primer lugar esos fragmentos menos comunes. De esta manera, dichos fragmentos se redistribuirán más rápidamente, consiguiendo que el número de copias de cada fragmento sea aproximadamente igual dentro del torrente.

Para determinar a qué solicitudes debe ella responder, BitTorrent utiliza un algoritmo de intercambio inteligente. La idea básica es que Alicia dé prioridad a los vecinos que actualmente están suministrando sus datos *a mayor velocidad*. Específicamente, para cada uno de los vecinos, Alicia mide de forma continua la velocidad a la que recibe bits y determina cuáles son los cuatro pares que le envían bits a mayor velocidad. Entonces, ella, de forma recíproca, envía fragmentos a esos mismos cuatro pares. Cada 10 segundos, vuelve a

calcular las velocidades y, posiblemente, tendrá que modificar el conjunto formado por los cuatro pares. En la jerga de BitTorrent, se dice que estos cuatro pares están **no filtrados** (*unchoked*). Además, y lo que es más importante, cada 30 segundos Alicia elige de forma aleatoria un vecino adicional y le envía fragmentos. Supongamos que el par elegido aleatoriamente es el de Benito. En la jerga de BitTorrent, se dice que Benito está **no filtrado de forma optimista**. Dado que Alicia está enviando datos a Benito, ella puede convertirse en uno de los cuatro suministradores principales de Benito, en cuyo caso Benito comenzaría a enviar datos a Alicia. Si la velocidad a la que Benito envía datos a Alicia es lo suficientemente alta, Benito podría entonces, a su vez, convertirse en uno de los cuatro suministradores principales de Alicia. En otras palabras, cada 30 segundos Alicia elegirá aleatoriamente un nuevo socio de intercambio e iniciará las transacciones con él. Si los dos pares están satisfechos con el intercambio, se incluirán en sus respectivas listas de los cuatro principales y continuarán realizando intercambios hasta que uno de los pares encuentre un socio mejor. El efecto es que los pares capaces de suministrar datos a velocidades compatibles tienden a emparejarse. La selección aleatoria de vecinos también permite a los nuevos pares obtener fragmentos, con el fin de tener algo que intercambiar. Todos los demás pares vecinos excepto estos cinco (los cuatro pares "principales" más el par de prueba) están "filtrados", es decir, no reciben fragmentos de Alicia. BitTorrent dispone de una serie de interesantes mecanismos que no vamos a ver aquí, entre los que se incluyen la gestión de piezas (minifragmentos), el procesamiento en cadena, la selección aleatoria del primer fragmento, el modo *endgame* y el *anti-snubbing* [Cohen 2003].

El mecanismo de incentivos para intercambio que acabamos de describir a menudo se denomina *tit-for-tat* (toma y daca, una estrategia de la teoría de juegos) [Cohen 2003]. Se ha demostrado que este esquema de incentivos puede soslayarse maliciosamente [Liogkas 2006; Locher 2006; Piatek 2007]. No obstante, el ecosistema BitTorrent ha tenido un éxito bárbaro, con millones de pares simultáneos compartiendo activamente archivos en cientos de miles de torrentes. Si BitTorrent se hubiera diseñado sin la estrategia *tit-for-tat* (o una variante), y aunque todo el resto de características fueran las mismas, es posible que BitTorrent no existiera actualmente, ya que la mayor parte de los usuarios hubieran pretendido aprovecharse de los demás [Saroiu 2002].

En [Guo 2005; Piatek 2007] se proponen interesantes variantes del protocolo BitTorrent. Además, muchas de las aplicaciones P2P para flujos multimedia en directo, como PPLive y ppstream, están inspiradas en BitTorrent [Hei 2007].

2.6.2 Tablas hash distribuidas (DHT)

Un componente crítico de muchas aplicaciones P2P y otras aplicaciones distribuidas es un índice (es decir, una base de datos simple) que soporte operaciones de búsqueda y de actualización. Cuando esta base de datos está distribuida, los pares pueden llevar a cabo operaciones de almacenamiento del contenido en caché y de enrutamiento complejo de consultas entre ellos. Dado que la indexación de la información y la realización de búsquedas es un componente crítico en estos sistemas, vamos a estudiar a continuación una técnica popular de indexación y de búsqueda: las tablas hash distribuidas (DHT, *Distributed Hash Table*).

Veamos cómo construir una base de datos distribuida simple sobre un gran número (del orden de millones) de pares que permita la indexación y la realización de consultas simples. La información almacenada en esta base de datos estará formada por parejas (clave, valor). Por ejemplo, las claves podrían ser los números de la seguridad social y los valores los nombres correspondientes de las personas; en este caso, una pareja (clave, valor) de ejemplo

sería (156-45-7081, Johnny Wu). Las claves también podrían ser nombres de contenidos (por ejemplo, nombres de películas, albums o paquetes software) y los valores podrían ser las direcciones IP en las que estuvieran almacenados dichos contenidos; por ejemplo, en este caso una pareja clave-valor sería (Led Zeppelin IV, 203.17.123.38). Los peers consultarán esta base de datos suministrando la clave: si existen parejas (clave, valor) en la base de datos que se corresponden con esa clave, entonces la base de datos devuelve los pares correspondientes al peer que haya realizado la consulta. Así, por ejemplo, si la base de datos almacena números de la seguridad social y los nombres de las personas asociados a ellos, un peer puede consultar un número concreto de la seguridad social y la base de datos le devolverá el nombre de la persona con ese número. Los pares también podrán insertar parejas (clave, valor) en la base de datos.

La construcción de una base de datos así es sencilla mediante una arquitectura cliente-servidor en la que todas las parejas (clave, valor) se encuentran almacenados en un servidor central. Este enfoque centralizado también se empleaba en los primeros sistemas P2P, como Napster. Pero el problema es significativamente más interesante y constituye un mayor desafío en sistemas distribuidos formados por millones de pares conectados sin una autoridad central. En un sistema P2P, lo que se desea es distribuir las parejas (clave, valor) a todos los pares, de manera que cada par sólo almacene un pequeño subconjunto de la totalidad de las parejas (clave, valor). Un método sencillo para construir una base de datos P2P consiste en (1) diseminar de manera aleatoria las parejas (clave, valor) a lo largo de los pares y (2) mantener en cada par una lista de las direcciones IP de todos los pares participantes. De esta manera, el par que realiza una consulta puede enviarla a todos los demás pares, y los pares que contienen las parejas (clave, valor) que se corresponden con la clave consultada pueden responder con las parejas apropiadas. Por supuesto, este método no es en absoluto escalable, ya que requeriría que cada par siguiera la pista a todos los demás pares (posiblemente millones de ellos) e, incluso peor, tendría que enviar cada consulta a *todos* los pares.

A continuación vamos a describir un método elegante para diseñar una base de datos P2P. Para ello, en primer lugar asignaremos un identificador a cada par, siendo cada identificador un entero comprendido en el rango [0, 2^n 1] para un cierto n fijo. Observe que cada identificador puede expresarse mediante una representación de n bits. También es necesario que cada clave sea un entero perteneciente al mismo rango. Si ha estado atento, se habrá dado cuenta de que las claves de ejemplo citadas anteriormente (los números de la seguridad social y los nombres de contenidos multimedia) no eran números enteros. Para generar enteros a partir de esas claves, vamos a utilizar una función hash que asocie cada clave (por ejemplo, el número de la seguridad social) a un número entero comprendido en el rango [0, 2^n 1]. Una función hash es una función muchos a uno para la que dos entradas diferentes pueden tener asociada la misma salida (el mismo entero), pero la probabilidad de que les corresponda esa misma salida es extremadamente baja. (Los lectores que no estén familiarizados con las funciones hash pueden consultar el Capítulo 7, en el que se estudian estas funciones en detalle.) Se asume que la función hash está públicamente disponible para todos los peers del sistema. De aquí en adelante, cuando hablemos de la "clave", estaremos haciendo referencia a la función hash de la clave original. Por ejemplo, si la clave original es "Led Zeppelin IV", la clave será el número entero igual al valor hash de "Led Zeppelin IV". Además, puesto que estamos empleando valores hash de claves, en lugar de las propias claves, de ahora en adelante nos referiremos a la base de datos distribuida como a una **tabla hash distribuida (DHT)**.

Abordemos el problema de almacenar las parejas (clave, valor) en la tabla DHT. La cuestión central es definir una regla para asignar las claves a los pares. Puesto que cada par

tiene asociado un identificador entero y cada clave también es un entero contenido en el mismo rango, sería natural asignar cada pareja (clave, valor) al peer cuyo identificador sea *el más próximo* a la clave. Para implementar este esquema, tendremos que definir qué queremos decir con "el más próximo", y para ello son muchos los convenios que podemos adoptar. Por conveniencia, vamos a definir el peer más próximo como el *inmediato sucesor de la clave.* Veamos ahora un ejemplo. Supongamos que n 4, luego en este caso todos los identificadores de los peers y las claves se encuentran en el rango [0, 15]. Supongamos también que hay ocho peers en el sistema con los siguientes identificadores: 1, 3, 4, 5, 8, 10, 12 y 15. Por último, supongamos que deseamos almacenar la pareja clave-valor (11, Johnny Wu) en uno de los ocho pares; pero ¿en qué par? Aplicando nuestro convenio del más próximo, dado que el par 12 es el inmediato sucesor para la clave 11, almacenaremos el par (11, Johnny Wu) en el par 12. Para completar nuestra definición del más próximo, establecemos que si la clave es exactamente igual a uno de los identificadores de par, almacenaremos la pareja (clave-valor) en dicho par; y si la clave es mayor que todos los identificadores de par, aplicaremos el convenio de módulo de 2^n, almacenando la pareja (clave-valor) en el par con el identificador más pequeño.

Imaginemos ahora que el par de Alicia desea insertar una pareja (clave-valor) en la tabla DHT. Conceptualmente, esto es sencillo: en primer lugar, ella determina el par cuyo identificador es más próximo a la clave; a continuación, envía un mensaje a dicho par, dándole instrucciones para almacenar la pareja (clave, valor). Pero, ¿cómo determina Alicia el par cuyo identificador es más próximo a la clave? Si Alicia tuviera que llevar la cuenta de todos los pares del sistema (sus identificadores y las correspondientes direcciones IP), podría determinar localmente cuál es el par más próximo. Pero este método requiere que *cada* par tenga controlados a *todos* los restantes pares de la tabla DHT, lo que es totalmente impracticable en un sistema a gran escala formado por millones de pares.

DHT circular

Para afrontar este problema de escala, consideremos que disponemos los pares en un círculo. Con esta disposición circular, cada par sólo tiene que controlar a su inmediato sucesor (módulo 2^n). En la Figura 2.27(a) se muestra un ejemplo de esta disposición circular. En este caso, n de nuevo es igual a 4 y hay exactamente los mismos ocho pares que en el ejemplo anterior. Cada par sólo es consciente de la existencia de su inmediato sucesor; por ejemplo, el par 5 conoce la dirección IP y el identificador del par 8, pero no tiene por qué saber nada acerca de los restantes pares que pueda haber en la DHT. Esta disposición circular de los pares es un caso especial de **red solapada**. En una red solapada, los pares forman una red lógica abstracta que reside por encima de la red de computadoras "subyacente" formada por los enlaces físicos, los routers y los hosts. Los enlaces de una red solapada no son enlaces físicos, simplemente son uniones virtuales entre parejas de pares. En la red solapada de la Figura 2.27(a), hay ocho pares y ocho enlaces solapados; en la red solapada de la Figura 2.27(b), hay ocho peers y 16 enlaces solapados. Normalmente, un enlace solapado utiliza muchos enlaces físicos y routers de la red subyacente.

Utilizando la red solapada circular de la Figura 2.27(a), suponemos que el par 3 desea determinar qué par de la tabla DHT es responsable de la clave 11, bien para insertar o consultar una pareja (clave, valor). Utilizando la red solapada, el par de origen (el par 3) crea un mensaje que dice: "¿Quién es el responsable de la clave 11?" y envía este mensaje a su sucesor, el par 4. Cuando un par recibe tal mensaje, dado que conoce el identificador de su sucesor, puede determinar si él es el responsable de esa clave en cuestión (es decir, el más

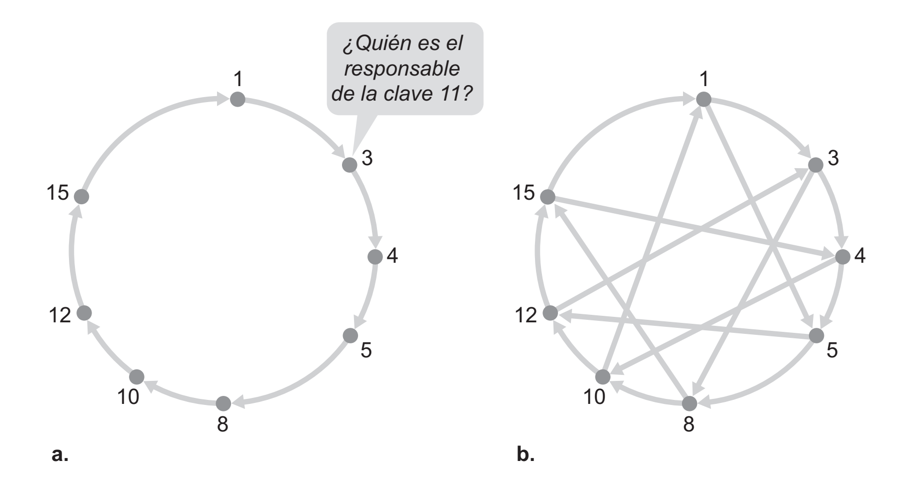


Figura 2.27 • (a) Una DHT circular. El peer 3 desea determinar quién es el responsable de la clave 11. (b) Una DHT circular con atajos.

próximo). Si un par no es el responsable de la clave, simplemente envía el mensaje a su sucesor. De este modo, por ejemplo, cuando el par 4 recibe el mensaje que pregunta por la clave 11, determina que no es el responsable de la misma (porque su sucesor está más próximo a la clave), por lo que simplemente pasa el mensaje a su propio sucesor, al par 5. Este proceso continúa hasta que el mensaje llega al par 12, el cual determina que él es el par más próximo a la clave 11. En este punto, el par 12 puede devolver un mensaje al origen, al par 3, informando de que es el responsable de la clave 11.

La DHT circular proporciona una solución muy elegante para reducir la cantidad de información solapada que debe gestionar cada par. En resumen, cada par sólo es consciente de la existencia de otros dos pares, su inmediato sucesor y su inmediato predecesor. (Por defecto, el par sabe de la existencia de su predecesor porque éste le envía mensajes.). Pero esta solución introduce un nuevo problema. Aunque cada par únicamente es consciente de sus dos pares vecinos, para encontrar al nodo responsable de una clave (en el caso peor), todos los *N* nodos de la DHT tendrán que reenviar un mensaje siguiendo el contorno del círculo, luego, como media, se enviarán *N*/2 mensajes.

Por tanto, al diseñar una DHT hay que llegar a un compromiso entre el número de vecinos del que cada par puede llevar la cuenta y el número de mensajes que la DHT necesita enviar para resolver una única consulta. Por un lado, si cada par lleva la cuenta de todos los demás pares (solapamiento en malla), entonces sólo habrá que enviar un mensaje por consulta, pero cada par tiene que controlar a *N* pares. Por otro lado, con una DHT circular, cada par sólo es consciente de la existencia de dos pares, pero se envían *N*/2 mensajes como media por cada consulta. Afortunadamente, podemos ajustar nuestro diseño de la DHT de manera que el número de vecinos por par, así como el número de mensajes por consulta se mantenga en un tamaño aceptable. Una posible mejora consistiría en utilizar la red solapada circular como base y añadir "atajos", de modo que cada par no sólo tenga controlado a su inmediato sucesor, sino también a un número relativamente pequeño de pares de "atajo" dispersos alrededor del círculo. En la Figura 2.27(b) se muestra un ejemplo de una DHT circular con algunos atajos. Estos atajos se emplean para agilizar el enrutamiento de los mensajes de consulta. Específicamente, cuando un par recibe un mensaje para consultar

una clave, reenvía el mensaje al vecino (al sucesor o a uno de los vecinos atajo) más próximo a la clave. Así, en la Figura 2.27(b), cuando el par 4 recibe el mensaje preguntando por la clave 11, determina que el par más próximo a la clave (entre sus vecinos) es su vecino atajo 10 y, a continuación, reenvía el mensaje directamente al par 10. Evidentemente, los atajos pueden reducir de forma significativa el número de mensajes utilizado para procesar una consulta.

En consecuencia, ahora la pregunta que debemos plantearnos es: "¿Cuántos vecinos atajo debe tener cada par y qué pares deberían ser esos vecinos atajo?" Los investigadores han dedicado mucho esfuerzo a resolver esta cuestión [Stoica 2001; Rowstron 2001; Ratnasamy 2001; Zhao 2004; Maymounkov 2002; Garces-Erce 2003]. Se ha demostrado que una DHT se puede diseñar de modo que tanto el número de vecinos por par, así como el número de mensajes por consulta, sea igual a O(log *N*), donde *N* es el número de pares. Este tipo de diseño proporciona un compromiso satisfactorio entre las soluciones extremas de emplear topologías solapadas de red en malla y redes circulares.

Abandono de los pares

En los sistemas P2P, un par puede entrar o salir del sistema sin avisar. Así, al diseñar una DHT también tenemos que preocuparnos por actualizar la red solapada DHT teniendo en cuenta el fenómeno del abandono de los pares. Con el fin de proporcionar una imagen general de cómo se podría conseguir esto, consideremos de nuevo la DHT circular de la Figura 2.27(a). Para gestionar el abandono de los pares, tendremos que hacer que cada par tenga controlados a su primer y segundo sucesores (es decir, que conozca sus direcciones IP); por ejemplo, ahora el par 4 tendrá que tener localizados a los pares 5 y 8. También necesitaremos que cada par verifique periódicamente que sus dos sucesores están activos (por ejemplo, enviándoles periódicamente mensajes ping y solicitándoles respuestas). Consideremos ahora cómo se actualiza una DHT cuando un par abandona repentinamente el sistema. Por ejemplo, supongamos que el par 5 de la Figura 2.27(a) abandona de repente la red. En este caso, los dos pares anteriores (4 y 3) detectan que el 5 ha abandonado porque ya no responde a sus mensajes ping. Los pares 4 y 3 tienen entonces que actualizar la información de estado que indica cuáles son sucesores. Veamos cómo el par 4 actualiza esta información de estado:

1. El par 4 sustituye a su primer sucesor (par 5) por su segundo sucesor (par 8).
2. A continuación, el par 4 pide a su nuevo primer sucesor (par 8) el identificador y la dirección IP de su sucesor inmediato (par 10). Después, el par 4 hace que el par 10 sea su segundo sucesor.

En los problemas de repaso deberá determinar cómo actualizaría el par 3 su información de enrutamiento para la red solapada.

Ahora que hemos visto lo que ocurre cuando un par abandona la red, consideremos lo que pasa cuando un par se une a la DHT. Supongamos que un par cuyo identificador es 13 desea unirse a la DHT y que en el momento de hacerlo sólo sabe de la existencia del par 1 en esa DHT. El par 13 debería enviar en primer lugar un mensaje al par 1, preguntando cuáles serán su predecesor y su sucesor. Este mensaje será reenviado a través de la DHT hasta llegar al par 12, quien se dará cuenta de que es el predecesor de 13 y de que su sucesor actual, el par 15, pasará a ser el sucesor de 13. A continuación, el par 12 envía esta información acerca del predecesor y del sucesor al par 13, el cual ahora puede unirse a la DHT haciendo al par 15 su sucesor y notificando al par 12 que debe modificar su sucesor inmediato a 13.

Las DHT se utilizan ampliamente en la práctica. Por ejemplo, BitTorrent utiliza la DHT Kademlia para crear un tracker distribuido. En BitTorrent, la clave es el identificador del torrente y el valor son las direcciones IP de todos los pares que están participando actualmente en el torrente [Falkner 2007, Neglia 2007]. De este modo, al consultar la DHT con un identificador de torrente, un par BitTorrent recién llegado puede determinar cuál es el par responsable del identificador (es decir, quién se encarga de llevar la cuenta de los pares existentes en el torrente). Una vez que ha localizado dicho par, el par recién llegado puede consultarle para obtener la lista de los demás pares del torrente. Las DHT se emplean ampliamente en el sistema de compartición de archivos eMule para la localización de contenido en los pares [Liang 2006].

2.6.3 Caso de estudio: telefonía Internet P2P con Skype

Skype es una aplicación P2P inmensamente popular, que suele tener del orden de siete u ocho millones de usuarios conectados simultáneamente. Además de proporcionar un servicio de telefonía Internet PC a PC, Skype ofrece servicios de telefonía de PC a teléfono, de teléfono a PC y de videoconferencia PC a PC. Fundada por los mismos emprendedores que crearon FastTrack y Kazaa, Skype fue adquirida por eBay en 2005 por 2.600 millones de dólares.

Skype utiliza técnicas P2P en una serie de formas bastante innovadoras, que ilustran a la perfección cómo puede utilizarse P2P en aplicaciones que van más allá de la distribución de contenido y la compartición de archivos. Al igual que sucede con la mensajería instantánea, la telefonía Internet PC a PC es inherentemente P2P, puesto que la base de la aplicación está en que hay parejas de usuarios (es decir, peers) comunicándose entre sí en tiempo real. Pero Skype también emplea técnicas P2P para otras dos importantes funciones: la ubicación de los usuarios y el NAT transversal.

No sólo son propietarios los protocolos de Skype sino que, además, todas las transmisiones de paquetes de Skype (paquetes de voz y de control) están cifradas. A pesar de ello, a partir de la información contenida en el sitio web de Skype y de una serie de estudios técnicos, los investigadores han logrado determinar cómo funciona Skype en términos generales [Baset 2006; Guha 2006; Chen 2006; Suh 2006; Ren 2006]. Al igual que sucede en FastTrack, los nodos de Skype están organizados en una red jerárquica superpuesta en la que cada par se clasifica como super par o como par normal. Skype incluye un índice que asigna los nombres de usuario de Skype a las direcciones IP actuales (junto con sus números de puertos). Este índice está distribuido sobre los super pares. Cuando Alicia quiere llamar a Benito, su cliente Skype busca en el índice distribuido para determinar la dirección IP actual de Benito. Puesto que el protocolo de Skype es propietario, no está claro actualmente cómo están organizadas las asignaciones de índice entre los super pares, aunque es bastante posible que se emplee algún tipo de organización DHT.

También se emplean técnicas P2P en los **retransmisores *(relays)*** Skype, que se usan para establecer llamadas entre hosts situados en redes domésticas. Muchas configuraciones de redes domésticas proporcionan acceso a Internet a través de un router (normalmente de un router inalámbrico). Estos routers son, en realidad, algo más que simples routers, y suelen incluir un mecanismo de Traducción de direcciones de red (NAT, *Network Address Translator*). Estudiaremos los mecanismos NAT en el Capítulo 4. Por ahora, lo único que necesitamos saber es que un mecanismo NAT impide que un host situado fuera de la red

doméstica inicie una conexión con un host situado dentro de esa red. Si *ambos* interlocutores Skype tienen activados mecanismos NAT, entonces se producirá un problema: ninguno de los dos podrá aceptar una llamada iniciada por el otro, lo que aparentemente hace que las llamadas sean imposibles. La utilización inteligente de los super pares y de los retransmisores permite resolver de forma elegante este problema. Suponga que en el momento de iniciar Alicia la sesión, se le asigna un super par sin mecanismo NAT. Alicia podrá iniciar una sesión con su super par, ya que su NAT sólo prohíbe las sesiones iniciadas fuera de su red doméstica. Esto permite a Alicia y a su super par intercambiar mensajes de control en esa sesión. Lo mismo sucede para Benito cuando inicia su sesión. Ahora, cuando Alicia quiere llamar a Benito informa a su super par, que a su vez informa al super par de Benito, que por su parte informa a Benito acerca de la llamada entrante de Alicia. Si Benito acepta la llamada, los dos super pares seleccionan un tercer super par sin NAT (el nodo retransmisor) cuyo trabajo consistirá en retransmitir los datos entre Alicia y Benito. Los super pares de Benito y Alicia dan entonces instrucciones a Benito y Alicia, respectivamente, para iniciar una sesión con el retransmisor. Alicia enviará sus paquetes de voz al retransmisor a través de la conexión Alicia-retransmisor (que ha sido iniciada por Alicia) y el retransmisor reenviará esos paquetes a través de la conexión retransmisor-Benito (que ha sido iniciada por Benito); los paquetes enviados por Benito a Alicia fluyen a través de las dos mismas conexiones de retransmisión, pero en orden inverso. ¡Problema resuelto! Benito y Alicia dispondrán de una conexión terminal a terminal bajo demanda aun cuando ninguno de los dos pueda aceptar una sesión que tenga su origen fuera de su red LAN. La utilización de retransmisores ilustra el diseño cada vez más sofisticado de los sistemas P2P, en los que los pares llevan a cabo servicios básicos del sistema para otros (dos ejemplos serían el servicio de índice y la retransmisión), al mismo tiempo que utilizan el servicio de usuario final (por ejemplo, la descarga de archivos o la telefonía IP) que el sistema P2P proporciona.

Skype es una aplicación Internet con un extraordinario éxito, siendo utilizada por decenas de millones de usuarios. La enormemente rápida adopción global de Skype, al igual que la de la compartición de archivos P2P, la Web y la mensajería instantánea anteriormente, es una prueba palpable de la sabiduría inherente al diseño arquitectónico global de Internet, un diseño que no podía haber previsto el enorme y cada vez más amplio conjunto de aplicaciones Internet que se desarrollarían a lo largo de los 30 años siguientes. Los servicios de red ofrecidos a las aplicaciones Internet [transporte de datagramas sin conexión (UDP), transferencia fiable de datagramas orientada a conexión (TCP), la interfaz de sockets, el direccionamiento y el sistema de nombres (DNS), entre otros], han demostrado ser capaces de permitir el desarrollo de miles de aplicaciones. Puesto que todas estas aplicaciones se han construido sobre las cuatro capas inferiores de la pila de protocolos de Internet, sólo necesitan que se desarrollen nuevos programas cliente-servidor y peer-to-peer para emplearlos en los sistemas de usuario final. Esto, a su vez, ha permitido que estas aplicaciones se implanten y adopten también de una forma extraordinariamente rápida.

2.7 Programación de sockets con TCP

Ahora que hemos examinado una serie de importantes aplicaciones de red, vamos a ver cómo se escriben en la práctica los programas de aplicaciones de redes. En esta sección, escribiremos programas de aplicación que utilizan TCP, mientras que en la sección siguiente veremos cómo se escriben los programas que usan UDP.

Recuerde de la Sección 2.1 que muchas aplicaciones de red están compuestas por una pareja de programas (un programa cliente y un programa servidor) que residen en dos sistemas terminales distintos. Cuando se ejecutan estos dos programas, se crean un proceso cliente y un proceso servidor, y estos dos procesos se comunican entre sí leyendo y escribiendo en sockets. Cuando se crea una aplicación de red, la tarea principal del desarrollador es escribir el código para los programas cliente y servidor.

Existen dos tipos de aplicaciones de red. Uno de ellos es una implementación de un estándar de protocolo definido en, por ejemplo, un RFC. Para este tipo de implementaciones, los programas cliente y servidor deben adaptarse a las reglas dictadas por ese RFC. Por ejemplo, el programa cliente podría ser una implementación del lado del cliente del protocolo FTP, descrito en la Sección 2.3 y definido explícitamente en el documento RFC 959; de forma similar, el programa servidor podría ser una implementación del protocolo de servidor FTP, que también está definido explícitamente en el documento RFC 959. Si un desarrollador escribe código para el programa cliente y otro desarrollador independiente escribe código para el programa servidor y ambos desarrolladores siguen cuidadosamente las reglas marcadas en el RFC, entonces los dos programas serán capaces de interoperar. Ciertamente, muchas de las aplicaciones de red actuales implican la comunicación entre programas cliente y servidor que han sido creados por desarrolladores independientes (por ejemplo, un navegador Firefox comunicándose con un servidor web Apache, o un cliente FTP de un PC cargando un archivo en un servidor FTP Linux). Cuando un programa cliente o un programa servidor implementa un protocolo definido en un RFC, debe emplear el número de puerto asociado con el protocolo. (Los números de puerto se han explicado brevemente en la Sección 2.1 y se tratan más detalladamente en el Capítulo 3.)

El otro tipo de aplicación de red son las aplicaciones propietarias. En este caso, el protocolo de la capa de aplicación utilizado por los programas cliente y servidor no tiene que cumplir necesariamente ninguna recomendación RFC existente. Un único desarrollador (o un equipo de desarrollo) crea tanto el programa cliente como el programa servidor, y ese desarrollador tiene el control completo sobre aquello que se incluye en el código. Pero como el código no implementa ningún protocolo de dominio público, otros desarrolladores independientes no podrán desarrollar código que interopere con esa aplicación. A la hora de crear una aplicación propietaria, el desarrollador debe tener cuidado de no utilizar ninguno de los números de puerto bien conocidos definidos en los documentos RFC.

En esta sección y en la siguiente vamos a examinar los problemas fundamentales del desarrollo de aplicaciones propietarias cliente-servidor. Durante la fase de desarrollo, una de las primeras decisiones que el desarrollador debe tomar es si la aplicación se ejecutará sobre TCP o sobre UDP. Recuerde que TCP está orientado a la conexión y proporciona un canal fiable de flujo de bytes a través del cual se transmiten los datos entre los dos sistemas terminales. Por su parte, UDP es un protocolo sin conexión, que envía paquetes de datos independientes de un sistema terminal a otro, sin ningún tipo de garantía acerca de la entrega.

En esta sección vamos a desarrollar una aplicación cliente simple que se ejecute sobre TCP. En la siguiente sección desarrollaremos otra aplicación cliente simple que se ejecute sobre UDP. Presentamos estas aplicaciones TCP y UDP en Java. Podríamos haber escrito el código en C o C++, pero hemos optado por Java principalmente porque en este lenguaje las aplicaciones se escriben de forma más clara y nítida. Con Java se utilizan menos líneas de código y cada línea puede explicarse a un programador inexperto sin demasiada dificultad. Pero no tiene por qué asustarse si no está muy familiarizado con Java. Si tiene experiencia de programación en cualquier otro lenguaje, debería poder seguir el código fácilmente.

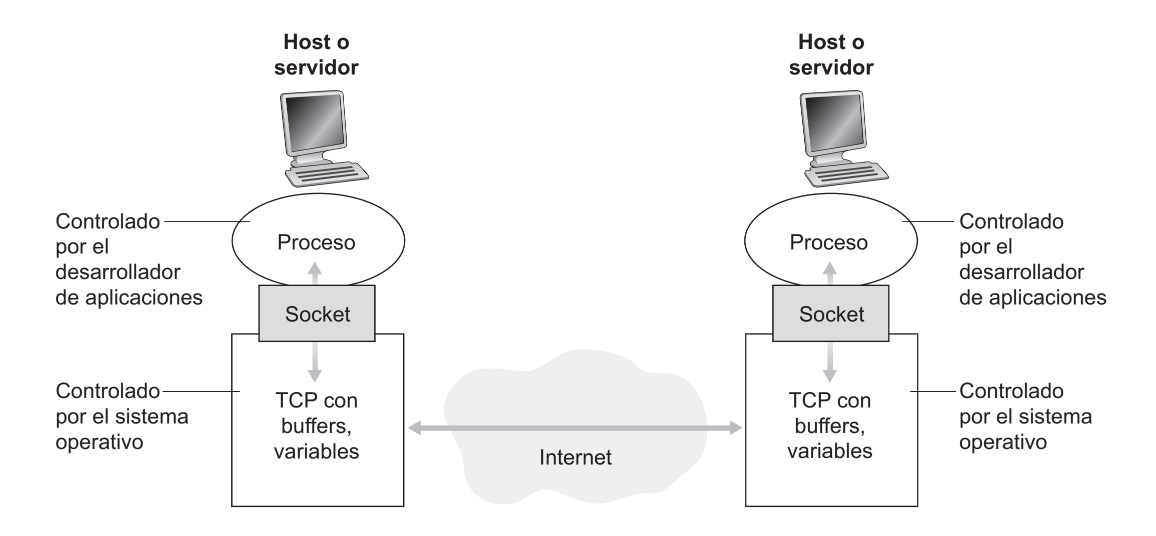


Figura 2.28 • Procesos comunicándose a través de sockets TCP.

Para los lectores que estén interesados en la programación cliente-servidor en C, hay disponibles varias buenas referencias [Donahoo 2001; Stevens 1997; Frost 1994; Kurose 1996].

2.7.1 Programación de sockets con TCP

Recuerde de la Sección 2.1 que los procesos que se ejecutan en máquinas diferentes se comunican entre sí enviando mensajes a través de sockets. Dijimos que cada proceso era análogo a una vivienda y que el socket del proceso era análogo a una puerta. Como se muestra en la Figura 2.28, el socket es la puerta entre el proceso de aplicación y TCP. El desarrollador de la aplicación dispone de control sobre todo lo que está situado en el lado de la capa de aplicación del socket; sin embargo, el control que tiene sobre el lado de la capa de transporte es muy pequeño (como mucho, el desarrollador de la aplicación tiene la posibilidad de fijar unos pocos parámetros de TCP, como el tamaño máximo de buffer y el tamaño máximo de segmento).

Ahora, examinemos más en detalle la interacción entre los programas cliente y servidor. Al cliente le corresponde iniciar el contacto con el servidor. Para que éste puede reaccionar al contacto inicial del cliente, tendrá que estar preparado, lo que implica dos cosas. En primer lugar, el programa servidor no puede estar durmiendo (es decir, tiene que estar ejecutándose como proceso antes de que el cliente trate de iniciar el contacto); en segundo lugar, el programa servidor debe disponer de algún tipo de puerta (o, más precisamente, un socket) que acepte algún contacto inicial procedente de un proceso cliente que se esté ejecutando en un host arbitrario. Utilizando nuestra analogía de la vivienda/puerta para un proceso/socket, en ocasiones nos referiremos a este contacto inicial del cliente diciendo que es equivalente a "llamar a la puerta de entrada".

Con el proceso servidor ejecutándose, el proceso cliente puede iniciar una conexión TCP con el servidor. Esto se hace en el programa cliente creando un socket. Cuando el cliente crea su socket, especifica la dirección del proceso servidor, es decir, la dirección IP

del host servidor y el número de puerto del proceso servidor. Una vez creado el socket en el programa cliente, el protocolo TCP del cliente inicia un proceso de acuerdo en tres fases y establece una conexión con el servidor. El proceso de acuerdo en tres fases, que tiene lugar en la capa de transporte, es completamente transparente para los programas cliente y servidor.

Durante el proceso de acuerdo en tres fases, el proceso cliente llama a la puerta de entrada del proceso servidor. Cuando el servidor "escucha" la llamada, crea una nueva puerta (o de forma más precisa, un nuevo socket) que estará dedicado a ese cliente concreto. En el ejemplo que sigue, nuestra puerta de entrada es un objeto `ServerSocket` que denominamos `socketAcogida`. Cuando un cliente llama a esta puerta, el programa invoca el método `accept()` de `socketAcogida`, que crea una nueva puerta para el cliente. Al final de la fase de negociación, existirá una conexión TCP entre el socket del cliente y el nuevo socket del servidor. En lo sucesivo, nos referiremos al nuevo socket dedicado del servidor con el nombre de **socket de conexión** del servidor.

Desde la perspectiva de la aplicación, la conexión TCP es un conducto virtual directo entre el socket del cliente y el socket de conexión del servidor. El proceso cliente puede enviar bytes arbitrarios a través de su socket, y TCP garantiza que el proceso servidor recibirá (a través del socket de conexión) cada byte en el orden en que ha sido enviado. Por tanto, TCP proporciona un **servicio fiable de flujo de bytes** entre los procesos cliente y servidor. Además, al igual que las personas pueden entrar y salir a través de una misma puerta, el proceso cliente no sólo envía bytes a través de su socket, sino que también puede recibirlos; de forma similar, el proceso servidor no sólo puede recibir, sino también enviar bytes a través de su socket de conexión. Esto se ilustra en la Figura 2.29. Puesto que los sockets desempeñan un papel fundamental en las aplicaciones cliente-servidor, el desarrollo de este tipo de aplicaciones también se suele denominar programación de sockets.

Antes de proporcionar un ejemplo de aplicación cliente-servidor, resulta útil explicar el concepto de flujo. Un **flujo** es una secuencia de caracteres que entran o salen de un proceso. Cada flujo puede ser un **flujo de entrada** o un **flujo de salida** del proceso. Si se trata de un flujo de entrada, entonces estará asociado a algún tipo de fuente de entrada para el proceso, como por ejemplo la entrada estándar (el teclado) o un socket a través del cual fluyan datos procedentes de Internet. Si se trata de un flujo de salida, entonces estará asociado con algún dispositivo de salida del proceso, como la salida estándar (el monitor) o un socket a través del cual fluyan los datos hacia Internet.

2.7.2 Ejemplo de aplicación cliente-servidor en Java

Vamos a utilizar la siguiente aplicación cliente-servidor simple para demostrar cómo programar un socket tanto para TCP como para UDP:

1. Un cliente lee una línea de su **entrada estándar** (teclado) y la envía a través de su socket al servidor.
2. El servidor lee una línea de su socket de conexión.
3. El servidor pasa la línea a caracteres en mayúscula.
4. El servidor envía la línea modificada a través de su socket de conexión al cliente.
5. El cliente lee la línea modificada de su socket y la muestra en su **salida estándar** (monitor).

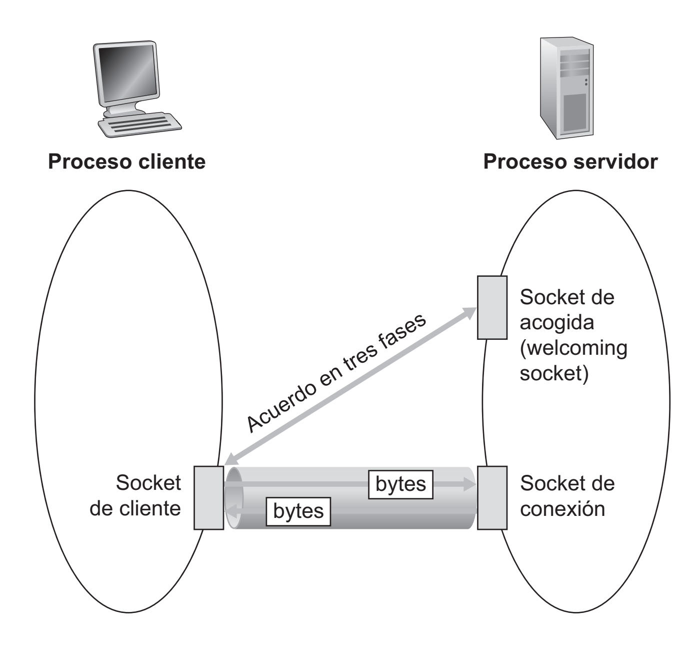


Figura 2.29 • Socket de cliente, socket de acogida y socket de conexión.

La Figura 2.30 ilustra la actividad relativa a los sockets del cliente y del servidor.

A continuación proporcionamos la pareja de programas cliente-servidor para una implementación TCP de la aplicación. Realizaremos un análisis detallado línea a línea de cada uno de los programas. El nombre del programa cliente es `TCPCliente.java` y el nombre del programa servidor es `TCPServidor.java`. Con el fin de poner el énfasis en las cuestiones fundamentales, hemos proporcionado de manera intencionada código que funciona correctamente pero que no es a prueba de balas. Un "código realmente bueno" tendría unas pocas más líneas auxiliares.

Una vez que los dos programas están compilados en sus respectivos hosts, el programa servidor se ejecuta primero en el host servidor, que crea un proceso servidor en el host. Como se ha mencionado anteriormente, el proceso servidor espera a ser contactado por un proceso cliente. En esta aplicación de ejemplo, cuando el programa cliente se ejecuta se crea un proceso en el cliente y este proceso contacta inmediatamente al servidor y establece una conexión TCP con él. El usuario que está en el cliente puede entonces utilizar la aplicación para enviar una línea y recibir a continuación la misma línea escrita en letras mayúsculas.

TCPCliente.java

He aquí el código para el lado del cliente de la aplicación:

```
import java.io.*;
import java.net.*;
class TCPCliente {
      public static void main(String argv[]) throws Exception
      {
         String frase;
```

```
        String fraseModificada;
        BufferedReader entradaDesdeUsuario = new BufferedReader(
           new InputStreamReader(System.in));
        Socket socketCliente = new Socket("nombrehost", 6789);
        DataOutputStream salidaAServidor = new DataOutputStream(
           socketCliente.getOutputStream());
        BufferedReader entradaDesdeServidor =
           new BufferedReader(new InputStreamReader(
              socketCliente.getInputStream()));
        frase = entradaDesdeUsuario.readLine();
        salidaAServidor.writeBytes(frase + '\n');
        modifiedSentence = entradaDesdeServidor.readLine();
        System.out.println("DEL SERVIDOR: " +
              fraseModificada);
        socketCliente.close();
     }
}
```

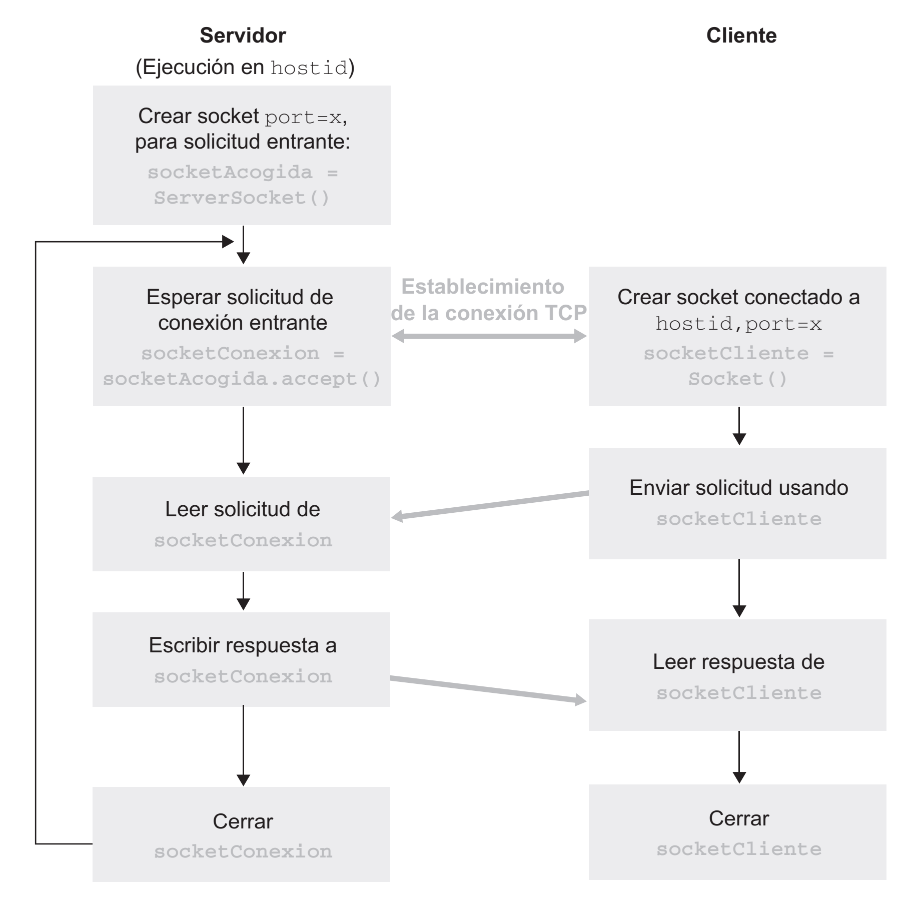


Figura 2.30 • Aplicación cliente-servidor que utiliza servicios de transporte orientados a la conexión.

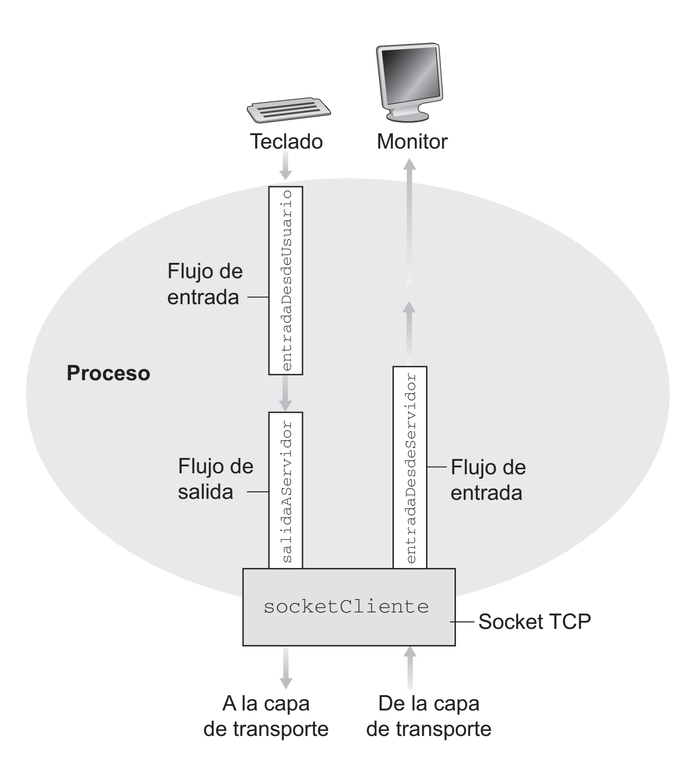


Figura 2.31 • `TCPcliente` tiene tres flujos a través de los cuales se transmiten los caracteres.

El programa `TCPCliente` crea tres flujos y un socket, como se muestra en la Figura 2.31. El socket se denomina `socketCliente`. El flujo `entradaDesdeUsuario` es un flujo de entrada al programa; está asociado a la entrada estándar (es decir, al teclado). Cuando el usuario escribe caracteres en el teclado, los caracteres entran en el flujo `entradaDesdeUsuario`. El flujo `entradaDesdeServidor` es otro flujo de entrada al programa y está asociado al socket. Los caracteres que llegan procedentes del flujo de la red entran en el flujo `entradaDesdeUsuario`. Por último, el flujo `salidaAServidor` es un flujo de salida del programa, que también está asociado al socket. Los caracteres que envía el cliente al flujo de la red entran en el flujo `salidaAServidor`.

Comentemos ahora las distintas líneas que forman el código.

```
import java.io.*;
import java.net.*;
```

`java.io` y `java.net` son paquetes Java. El paquete `java.io` contiene clases para los flujos de entrada y de salida. En concreto, el paquete `java.io` contiene las clases `BufferedReader` y `DataOutputStream`, clases que utiliza el programa para crear los tres flujos anteriormente mencionados. El paquete `java.net` proporciona clases para el soporte de red. En particular, contiene las clases `Socket` y `ServerSocket`. El objeto `socketCliente` de este programa se deriva de la clase `Socket`.

```
class TCPCliente {
      public static void main(String argv[]) throws Exception
            {......}
}
```

Hasta el momento, lo que hemos visto es lo mismo que puede encontrar al principio de la mayoría de los códigos en Java. La tercera línea es el principio de un bloque de definición de una clase. La palabra clave `class` inicia la definición de la clase cuyo nombre es `TCPCliente`. Una clase contiene variables y métodos. Tanto las variables como los métodos se encierran entre llaves que establecen el principio y el final del bloque de definición de la clase. La clase `TCPCliente` no tiene ninguna variable de clase y tiene exactamente un solo método, el método `main()`. Los métodos son similares a las funciones o los procedimientos en lenguajes como por ejemplo C; el método `main()` en el lenguaje Java es como la función `main()` de C y C++. Cuando el intérprete Java ejecuta una aplicación (al ser invocado para la clase que controla la aplicación), comienza llamando al método `main()` de la clase. Entonces, el método `main()` llama a los restantes métodos necesarios para ejecutar la aplicación. En esta introducción a la programación de sockets en Java, vamos a ignorar las palabras clave `public`, `static`, `void`, `main` y `throws Exceptions` (aunque debe incluirlas en el código).

```
String frase;
String fraseModificada;
```

Las dos líneas anteriores declaran objetos de tipo `String`. El objeto `frase` es la cadena escrita por el usuario y enviada al servidor. El objeto `fraseModificada` es la cadena proporcionada por el servidor y enviada a la salida estándar del usuario.

```
BufferedReader entradaDesdeUsuario = new BufferedReader(
        new InputStreamReader(System.in));
```

La línea anterior crea el objeto de flujo `entradaDesdeUsuario` de tipo `Buffered Reader`. El flujo de entrada se inicializa con `System.in`, que asocia el flujo a la entrada estándar. El comando permite al cliente leer texto introducido a través de su teclado.

```
Socket socketCliente = new Socket("nombrehost", 6789);
```

La línea anterior crea el objeto `socketCliente` de tipo `Socket`. También inicia la conexión TCP entre el cliente y el servidor. La cadena `"nombrehost"` tiene que sustituirse por el nombre de host del servidor (por ejemplo, `"apple.poly.edu"`). Antes de que la conexión TCP se inicie realmente, el cliente realiza una búsqueda DNS con el nombre de host para obtener la dirección IP correspondiente. El número 6789 es el número de puerto. Puede utilizar un número de puerto diferente, pero debe asegurarse de utilizar el mismo número de puerto en el lado del servidor de la aplicación. Como ya hemos mencionado con anterioridad, la dirección IP del host junto con el número de puerto de la aplicación identifican al proceso servidor.

```
DataOutputStream salidaAServidor =
      new DataOutputStream(socketCliente.getOutputStream());
BufferedReader entradaDesdeServidor =
      new BufferedReader(new inputStreamReader(
              socketCliente.getInputStream()));
```

Las líneas anteriores crean objetos de flujo que están asociados al socket. El flujo `salidaAServidor` proporciona la salida del proceso hacia el socket. El flujo `entradaDesdeServidor` proporciona la entrada de proceso a partir del socket (véase la Figura 2.31).

```
frase = entradaDesdeServidor.readLine();
```

Esta línea introduce el texto escrito por el usuario en la cadena `frase`. Esta cadena `frase` continúa recopilando caracteres hasta que el usuario termina la línea introduciendo un retorno de carro. La línea pasa desde la entrada estándar a través del flujo `entradaDesdeUsuario` a la cadena `frase`.

```
salidaAServidor.writeBytes(frase + '\n');
```

La línea anterior introduce la cadena `frase` con un retorno de carro adicional en el flujo `salidaAServidor`. Esta cadena aumentada fluye a través del socket del cliente y entra en el canal TCP. El cliente entonces espera a recibir los caracteres procedentes del servidor.

```
fraseModificada = entradaDesdeServidor.readLine();
```

Cuando llegan caracteres desde el servidor, entran en el flujo `entradaDesdeServidor` y se almacenan en la cadena `fraseModificada`. Los caracteres continúan acumulándose en `modifiedSentence` hasta que la línea termina con un carácter de retorno de carro.

```
System.out.println("DEL SERVIDOR " + fraseModificada);
```

La línea anterior muestra en el monitor la cadena `fraseModificada` devuelta por el servidor.

```
socketCliente.close();
```

Esta última línea cierra el socket y, por tanto, cierra también la conexión TCP entre el cliente y el servidor. Hace que TCP en el cliente envíe un mensaje TCP a TCP en el servidor (véase la Sección 3.5).

TCPServidor.java

Ahora veamos el programa de servidor.

```
import java.io.*;
import java.net.*;
class TCPServidor {
      public static void main(String argv[]) throws Exception
         {
            String fraseCliente;
            String fraseMayusculas;
            ServerSocket socketAcogida = new ServerSocket (6789);
            while(true) {
               Socket socketConexion = socketAcogida.accept();
               BufferedReader entradaDesdeCliente =
                  new BufferedReader(new InputStreamReader(
                       socketConexion.getInputStream()));
               DataOutputStream salidaACliente =
```

```
                    new DataOutputStream(
                         socketConexion.getOutputStream());
                fraseCliente = entradaDesdeCliente.readLine();
                fraseMayusculas =
                         fraseCliente.toUpperCase() + '\n';
                salidaACliente.writeBytes(fraseMayusculas);
            }
        }
}
```

`TCPServidor` tiene muchas similitudes con `TCPCliente`. Analicemos ahora las líneas de `TCPServidor.java`. No comentaremos las líneas que son idénticas o similares a los comandos del programa `TCPCliente.java`.

La primera línea de `TCPServidor` es completamente distinta a lo que hemos visto en `TCPCliente`:

```
ServerSocket socketAcogida  = new ServerSocket(6789);
```

Esta línea crea el objeto `socketAcogida`, cuyo tipo es `ServerSocket`. El objeto `socketAcogida` es como una puerta que escucha a ver si un cliente llama. `socketAcogida` escucha en el número de puerto 6789. La siguiente línea es:

```
Socket socketConexion = socketAcogida .accept();
```

Esta línea crea un *nuevo* socket, denominado `socketConexion`, cuando algún cliente envía una solicitud a través de `socketAcogida`. Este socket también tiene el número de puerto 6789. (En el Capítulo 3 explicaremos por qué los dos sockets tienen el mismo número de puerto.) TCP establece a continuación un canal virtual directo entre `socketCliente` en el cliente y `socketConexion` en el servidor. El cliente y el servidor pueden entonces enviarse bytes a través del canal y todos los bytes llegan al otro lado en orden. Una vez establecido `socketConexion`, el servidor puede continuar a la escucha de solicitudes procedentes de otros clientes y dirigidas a la aplicación, utilizando `socketAcogida`. (Esta versión del programa no se pone realmente a la escucha de otras solicitudes de conexión, pero se puede modificar utilizando hebras para que lo haga.) El programa crea a continuación varios objetos de flujo de datos, análogos a los objetos de flujo de datos de `socketCliente`. Fijémonos ahora en esta línea:

```
fraseMayusculas = fraseCliente.toUpperCase() + '\n';
```

Este comando es el núcleo de la aplicación. Toma la línea enviada por el cliente, la pasa a caracteres en mayúscula y añade un retorno de carro. Utiliza el método `toUpperCase()`. Los restantes comandos del programa son menos fundamentales; se emplean para la comunicación con el cliente.

Para probar los dos programas, instale y compile `TCPCliente.java` en un host y `TCPServidor.java` en otro. Asegúrese de incluir el nombre de host apropiado del servidor en `TCPCliente.java`. A continuación, ejecute en el servidor `TCPServidor.class`, el programa de servidor compilado. De este modo, se crea un proceso en el servidor que está inactivo hasta que es contactado por algún cliente. A continuación, ejecute en el cliente `TCPCliente.class`, el programa cliente compilado. Esto crea un proceso en el cliente y

establece una conexión TCP entre los procesos cliente y servidor. Por último, para utilizar la aplicación, escriba una cadena de caracteres seguida por un retorno de carro.

Para desarrollar su propia aplicación cliente-servidor, puede comenzar modificando ligeramente los programas. Por ejemplo, en lugar de poner todas las letras en mayúsculas, el servidor puede contar el número de veces que aparece la letra *s* y devolver dicho número.

2.8 Programación de sockets con UDP

Hemos visto en la sección anterior que cuando dos procesos se comunican mediante TCP, es como si existiera un conducto (*pipe*) entre los mismos. Este conducto se mantiene hasta que uno de los dos procesos se cierra. Cuando uno de los procesos desea enviar algunos bytes al otro, simplemente inserta los bytes en el conducto. El proceso de envío no asocia una dirección de destino a los bytes porque el conducto está conectado lógicamente al destino. Además, el conducto proporciona un canal fiable de flujo de bytes; la secuencia de bytes recibida por el proceso receptor es exactamente igual a la que el emisor insertó en el conducto.

UDP también permite que dos (o más) procesos que estén ejecutándose en hosts diferentes se comuniquen. Sin embargo, UDP se diferencia de TCP en muchos puntos fundamentales. En primer lugar, UDP es un servicio sin conexión (no existe una fase inicial de negociación durante la que se establezca un conducto para la comunicación entre los dos procesos). Dado que UDP no dispone de un conducto, cuando un proceso desea enviar un lote de bytes a otro proceso, el proceso emisor tiene que asociar la dirección del proceso de destino al lote de bytes. Y esto tiene que hacerse para cada lote de bytes que el proceso emisor desee transmitir. Veamos una analogía. Considere un grupo de 20 personas que toman cinco taxis para dirigirse a un mismo destino; según las personas van subiendo en los taxis informan al conductor sobre el lugar al que desean ir. Así, UDP es similar a un servicio de taxi. La dirección de destino es una tupla formada por la dirección IP del host de destino y el número de puerto del proceso de destino. Al conjunto formado por el lote de bytes de información, la dirección IP de destino y el número de puerto vamos a denominarlo "paquete". UDP proporciona un modelo de servicio no fiable orientado al mensaje, en el que se hace el máximo esfuerzo posible por suministrar el lote de bytes al destino. Es un servicio orientado al mensaje, en el sentido de que los lotes de bytes que se envían en una única operación en el lado del emisor serán entregados como un lote en el extremo de recepción; esto contrasta con la semántica de flujo de bytes de TCP. UDP es un servicio de entrega *de mejor esfuerzo (best effort)* en el que no se garantiza que el lote de bytes sea entregado. El servicio UDP por tanto se diferencia enormemente (en varios aspectos) del modelo de servicio fiable de flujo de bytes de TCP.

Una vez creado un paquete, el proceso emisor lo pone en la red a través de un socket. Continuando con nuestra analogía de los taxis, en el otro lado del socket emisor hay un taxi esperando al paquete. El taxi entonces se pone en marcha y se dirige a la dirección de destino del paquete. Sin embargo, el taxi no garantiza que finalmente el paquete llegue a la dirección de destino (el taxi se puede averiar o sufrir cualquier otro tipo de imprevisto). En otras palabras, *UDP proporciona un servicio de transporte no fiable a los procesos en comunicación* (no garantiza que un paquete llegue a su destino final).

En esta sección vamos a ilustrar la programación de sockets volviendo a desarrollar la misma aplicación que en la sección anterior, pero en esta ocasión para UDP. Compro-

baremos que el código para UDP es diferente del código para TCP en muchos puntos importantes. En particular, (1) no existe una fase de acuerdo inicial entre los dos procesos y, por tanto, no se necesita el socket de acogida, (2) no hay flujos asociados a los sockets, (3) el host emisor crea paquetes asociando la dirección IP de destino y el número de puerto a cada lote de bytes que desea enviar y (4) el proceso receptor tiene que desenmarañar cada paquete recibido para obtener los bytes de información del paquete. Recordemos una vez más nuestra sencilla aplicación:

1. Un cliente lee una línea de su entrada estándar (el teclado) y la envía a través de su socket al servidor.
2. El servidor lee la línea de su socket.
3. El servidor convierte la línea a mayúsculas.
4. El servidor envía la línea modificada al cliente a través de su socket.
5. El cliente lee la línea modificada en su socket y la muestra en su salida estándar (el monitor).

La Figura 2.32 destaca la actividad principal relativa a los sockets del cliente y del servidor que se comunican sobre un servicio de transporte sin conexión (UDP).

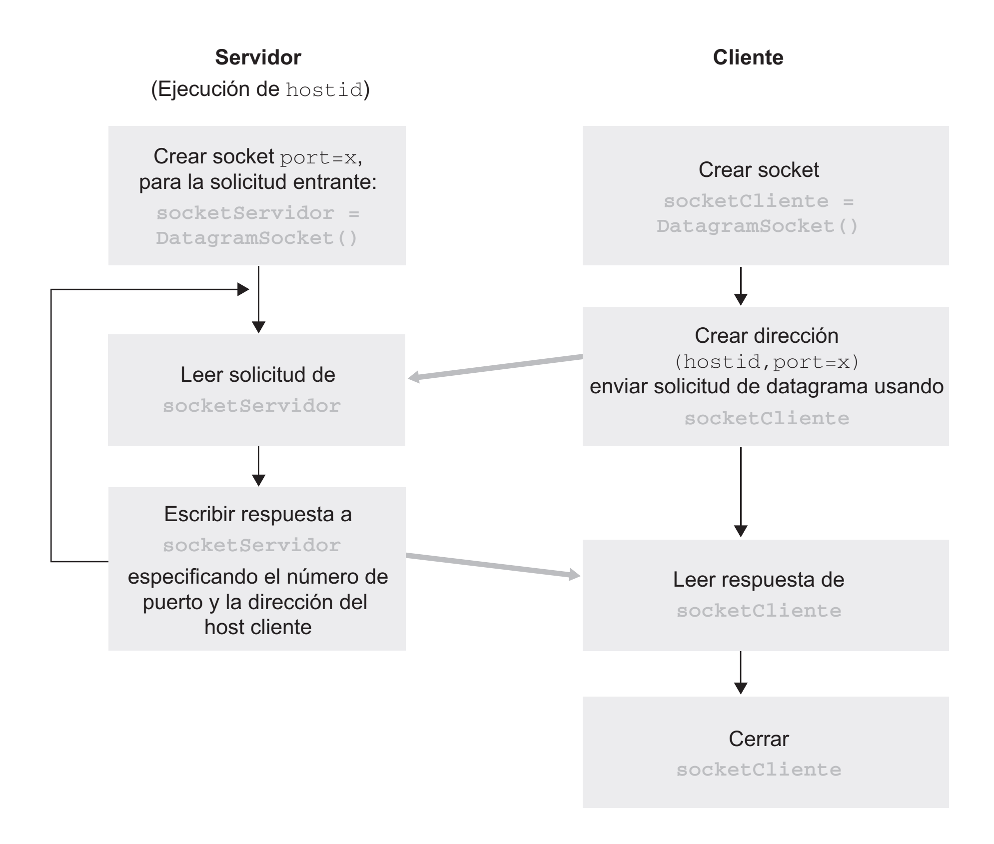


Figura 2.32 • Aplicación cliente-servidor que utiliza servicios de transporte sin conexión.

UDPCliente.java

He aquí el código para el lado del cliente de la aplicación:

```
import java.io.*;
import java.net.*;
class UDPCliente {
      public static void main(String args[]) throws Exception
      {
         BufferedReader entradaDesdeUsuario =
            new BufferedReader(new InputStreamReader
                              (System.in));
         DatagramSocket socketCliente= new DatagramSocket();
         InetAddress DireccionIP =
                              InetAddress.getByName("nombrehost");
         byte[] enviarDatos = new byte[1024];
         byte[] recibirDatos = new byte[1024];
         String frase = entradaDesdeUsuario.readLine();
         enviarDatos = frase.getBytes();
         DatagramPacket enviarPaquete =
            new DatagramPacket(enviarDatos, enviarDatos.length,
                              DireccionIP, 9876);
         socketCliente.send(enviarPaquete);
         DatagramPacket recibirPaquete =
            new DatagramPacket(recibirDatos,
                              recibirDatos.length);
         socketCliente.receive(recibirPaquete);
         String fraseModificada =
            new String(recibirPaquete.getData());
         System.out.println("DEL SERVIDOR:" +
                              fraseModificada);
         socketCliente.close();
      }
}
```

Como se muestra en la Figura 2.33, el programa `UDPCliente.java` crea un flujo y un socket. El nombre del socket es `socketCliente` y su tipo es `DatagramSocket`. Observe que UDP utiliza en el cliente una clase diferente de socket que TCP. En concreto, con UDP nuestro cliente utiliza el tipo `DatagramSocket`, mientras que con TCP nuestro cliente empleaba un `Socket`. El flujo `entradaDesdeUsuario` es un flujo de entrada al programa, que está conectado a la entrada estándar, es decir, al teclado. Teníamos un flujo equivalente en nuestra versión TCP del programa. Cuando el usuario escribe caracteres en el teclado, estos entran en el flujo `entradaDesdeUsuario`. Pero a diferencia de lo que ocurre en TCP, en este caso no hay flujos (ni de entrada ni de salida) asociados al socket. En lugar de introducir bytes en el flujo asociado a un objeto `Socket`, UDP introducirá paquetes individuales a través del objeto `DatagramSocket`.

Echemos ahora un vistazo a las líneas del código que son significativamente diferentes de las del programa `TCPCliente.java`.

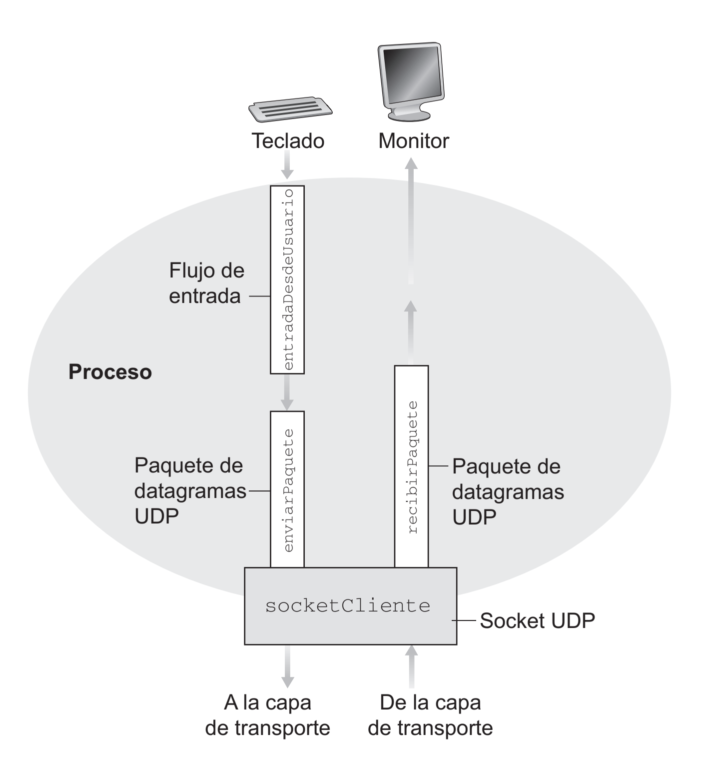


Figura 2.33 • `UDPClient` tiene un flujo; el socket acepta paquetes del proceso y suministra paquetes al proceso.

```
DatagramSocket socketCliente= new DatagramSocket();
```

Esta línea crea el objeto `socketCliente`de tipo `DatagramSocket`. En contraste con `TCPCliente.java`, esta línea no inicia una conexión TCP. En concreto, el cliente no se pone en contacto con el host servidor al ejecutar esta línea. Por esta razón, el constructor `DatagramSocket()` no toma como argumentos ni el nombre de host del servidor ni el número de puerto. Utilizando nuestra analogía de las puertas y conductos, la ejecución de la línea anterior crea una puerta para el proceso cliente, pero no crea un conducto entre los dos procesos.

```
InetAddress DireccionIP = InetAddress.getByName("nombrehost");
```

Para enviar bytes a un proceso de destino, necesitamos conocer la dirección del proceso. Parte de esta dirección es la dirección IP del host de destino. La línea anterior invoca una búsqueda DNS que traduzca el nombre de host (en este ejemplo, suministrado en el código por el desarrollador) en una dirección IP. La versión TCP del programa cliente también invocaba a DNS, aunque lo hacía de manera implícita, en lugar de explícita. El método `getByName()` toma como argumento el nombre de host del servidor y devuelve la dirección IP del mismo y almacena dicha dirección en el objeto `DireccionIP` de tipo `InetAddress`.

```
byte[] enviarDatos = new byte[1024];
byte[] recibirDatos = new byte[1024];
```

Los arrays de bytes `enviarDatos` y `recibirDatos` almacenarán los datos que el cliente envía y recibe, respectivamente.

```
enviarDatos = frase.getBytes();
```

La línea anterior realiza básicamente una conversión de tipos. Toma la cadena `frase` y la renombra como `enviarDatos`, que es un array de bytes.

```
DatagramPacket enviarPaquete = new DatagramPacket(
        enviarDatos, enviarDatos.length, DireccionIP, 9876);
```

Esta línea construye el paquete, `enviarPaquete`, que el cliente introducirá en la red a través de su socket. Este paquete incluye los datos contenidos en el paquete `enviarDatos`, la longitud de estos datos, la dirección IP del servidor y el número de puerto de la aplicación (que hemos definido como 9876). Observe que `enviarPaquete` es de tipo `DatagramPacket`.

```
socketCliente.send(enviarPaquete);
```

En la línea anterior, el método `send()` del objeto `socketCliente` toma el paquete que se acaba de crear y lo introduce en la red a través de `socketCliente`. Observe una vez más que UDP envía la línea de caracteres de forma muy diferente a como lo hace TCP. TCP simplemente insertaba la cadena de caracteres en un flujo de datos, que tenía una conexión lógica directa con el servidor; UDP crea un paquete que incluye la dirección del servidor. Después de enviar el paquete, el cliente espera entonces recibir un paquete procedente del servidor.

```
DatagramPacket recibirPaquete =
        new DatagramPacket(recibirDatos, recibirDatos.length);
```

En la línea anterior, el cliente crea un contenedor para el paquete, `recibirPaquete`, un objeto de tipo `DatagramPacket`, mientras espera recibir el paquete del servidor.

```
socketCliente.receive(recibirPaquete);
```

El cliente está inactivo hasta que recibe un paquete y cuando le llega, lo almacena en `recibirPaquete`.

```
String fraseModificada =
        new String(recibirPaquete.getData());
```

La línea anterior extrae los datos de `recibirPaquete` y realiza una conversión de tipos, convirtiendo un array de bytes en la cadena `fraseModificada`.

```
System.out.println("DEL SERVIDOR:" + fraseModificada);
```

Esta línea, que también aparece en `TCPCliente`, muestra la cadena `fraseModificada` en el monitor del cliente.

```
socketCliente.close();
```

Esta última línea cierra el socket. Puesto que UDP proporciona un servicio sin conexión, esta línea no hace que el cliente envíe un mensaje de la capa de transporte al servidor (a diferencia del caso de `TCPCliente`).

UDPServidor.java

Veamos ahora el lado del servidor de la aplicación:

```
import java.io.*;
import java.net.*;
class UDPServidor {
      public static void main(String args[]) throws Exception
         {
         DatagramSocket socketServidor = new
                      DatagramSocket(9876);
         byte[] recibirDatos = new byte[1024];
         byte[] enviarDatos = new byte[1024];
         while(true)
            {
            DatagramPacket recibirPaquete =
                  new DatagramPacket(recibirDatos,
                            recibirDatos.length);
            socketServidor.receive(recibirPaquete);
            String frase = new String(
                            recibirPaquete.getData());
            InetAddress DireccionIP =
                            recibirPaquete.getAddress();
            int puerto = recibirPaquete.getPort();
            String fraseMayusculas =
                            frase.toUpperCase();
            enviarDatos = fraseMayusculas.getBytes();
            DatagramPacket enviarPaquete =
            new DatagramPacket(enviarDatos,
                      enviarDatos.length, DireccionIP, puerto);
            socketServidor.send(enviarPaquete);
         }
      }
}
```

El programa `UDPServidor.java` crea un socket, como se muestra en la Figura 2.34. El nombre del socket es `socketServidor`. Se trata de un objeto de tipo `DatagramSocket`, al igual que lo era el socket del lado del cliente de la aplicación. De nuevo, no hay flujos asociados al socket.

Examinemos las líneas de este código que son diferentes a las de `TCPServidor.java`.

```
DatagramSocket socketServidor = new DatagramSocket(9876);
```

La línea anterior crea el `socketServidor` de tipo `DatagramSocket` en el puerto 9876. Todos los datos enviados y recibidos pasarán a través de este socket. Puesto que UDP pro-

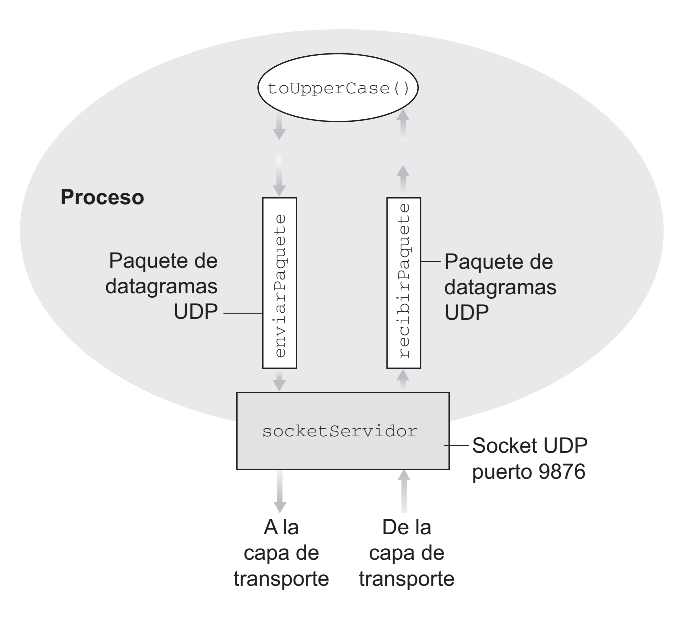


Figura 2.34 • `UDPServidor` no tiene flujos; el socket acepta paquetes del proceso y suministra paquetes al proceso.

porciona un servicio sin conexión, no tenemos que crear un nuevo socket y continuar escuchando para captar nuevas solicitudes de conexión como se hace en `TCPServidor.java`. Si varios clientes acceden a esta aplicación, todos ellos enviarán sus paquetes a través de esta misma puerta, `socketServidor`.

```
String frase = new String(recibirPaquete.getData());
InetAddress DireccionIP = recibirPaquete.getAddress();
int puerto = recibirPaquete.getPort();
```

Las tres líneas anteriores decodifican el paquete que llega procedente del cliente. La primera de ellas extrae los datos del paquete y los almacena en la cadena `frase`; hay una línea similar en el programa `UDPCliente`. La segunda línea extrae la dirección IP y la tercera línea extrae el número de puerto del cliente, que es elegido por el cliente y es diferente del número de puerto de servidor, 9876 (en el siguiente capítulo estudiaremos más en detalle los números de puerto del cliente). Necesariamente, el servidor tiene que obtener la dirección IP y el número de puerto del cliente, con el fin de poder devolverle el texto convertido a mayúsculas al cliente.

Esto completa nuestro análisis de los dos programas UDP. Para probar la aplicación, instale y compile `UDPCliente.java` en un host y `UDPServidor.java` en otro host. (Asegúrese de incluir el nombre de host apropiado del servidor en el programa `UDPCliente.java`.) A continuación, ejecute los dos programas en sus respectivos hosts. A diferencia de TCP, puede ejecutar primero el lado del cliente y luego el lado del servidor. La razón de ello es que el proceso cliente no intenta establecer una conexión con el servidor nada más ejecutarse el programa cliente. Una vez que haya ejecutado los programas cliente y servidor, puede utilizar la aplicación escribiendo una línea en el cliente.

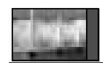
2.9 Resumen

En este capítulo hemos estudiado los aspectos conceptuales y de implementación de las aplicaciones de red. Hemos podido comprobar la omnipresencia de la arquitectura cliente-servidor adoptada por muchas aplicaciones de Internet y ver su uso en los protocolos HTTP, FTP, SMTP, POP3 y DNS. Hemos estudiado estos importantes protocolos del nivel de aplicación y sus correspondientes aplicaciones asociadas (la Web, la transferencia de archivos, el correo electrónico y DNS) con cierto detalle. También hemos aprendido acerca de la arquitectura P2P que está empezando a predominar y cómo se utiliza en muchas aplicaciones. Hemos examinado cómo puede utilizarse la API de sockets para crear aplicaciones de red. Asimismo, hemos estudiado el uso de los sockets para los servicios de transporte terminal a terminal orientados a la conexión (TCP) y sin conexión (UDP). ¡Hemos completado la primera etapa de nuestro viaje por la arquitectura de red en capas!

Al principio del libro, en la Sección 1.1, hemos proporcionado una definición algo vaga de protocolo: “el formato y el orden de los mensajes intercambiados entre dos o más entidades que se comunican, así como las acciones realizadas en la transmisión y/o recepción de un mensaje u otro evento.” El material facilitado en este capítulo, y en concreto el estudio detallado de los protocolos HTTP, FTP, SMTP, POP3 y DNS, aporta a esta definición una gran profundidad. Los protocolos son un concepto clave en las redes y nuestro estudio de los protocolos de aplicación nos ha proporcionado la oportunidad de desarrollar una idea más intuitiva sobre qué son los protocolos.

En la Sección 2.1 hemos descrito los modelos de servicio que ofrecen TCP y UDP a las aplicaciones que los invocan. En las Secciones 2.7 y 2.8 hemos visto en detalle estos modelos de servicio para el desarrollo de aplicaciones sencillas que se ejecutan sobre TCP y UDP. Sin embargo, hemos hablado poco acerca de cómo TCP y UDP proporcionan estos modelos de servicio. Por ejemplo, sabemos que TCP proporciona un servicio de datos fiable, pero todavía no sabemos cómo lo hace. En el siguiente capítulo veremos detenidamente no sólo *qué* servicios proporcionan, sino también el *cómo* y el *por qué* del funcionamiento de los protocolos de transporte.

Ahora que ya tenemos algunos conocimientos acerca de la estructura de las aplicaciones de Internet y de los protocolos de la capa de aplicación, estamos preparados para seguir descendiendo por la pila de protocolos y examinar la capa de transporte en el Capítulo 3.



Problemas y cuestiones de repaso

Capítulo 2 Cuestiones de repaso

SECCIÓN 2.1

R.1. Enumere cinco aplicaciones de Internet no propietarias y los protocolos de la capa de aplicación que utilizan.

R.2. ¿Cuál es la diferencia entre la arquitectura de red y la arquitectura de aplicación?

R.3. En una sesión de comunicación entre dos procesos, ¿qué proceso es el cliente y qué proceso es el servidor?

R.4. En una aplicación de compartición de archivos P2P, ¿está de acuerdo con la siguiente afirmación: "No existen los lados de cliente y de servidor en una sesión de comunicación"? ¿Por qué?

R.5. ¿Qué información utiliza un proceso que se ejecuta en un host para identificar a un proceso que se ejecuta en otro host?

R.6. Suponga que desea realizar una transición desde un cliente remoto a un servidor lo más rápidamente posible. ¿Qué utilizaría, UDP o TCP? ¿Por qué?

R.7. Utilizando la Figura 2.4, podemos ver que ninguna de las aplicaciones indicadas en dicha figura presenta a la vez requisitos de temporización y de ausencia de pérdida de datos. ¿Puede concebir una aplicación que requiera que no haya pérdida de datos y que también sea extremadamente sensible al tiempo?

R.8. Enumere las cuatro clases principales de servicios que puede proporcionar un protocolo de transporte. Para cada una de las clases de servicios, indique si UDP o TCP (o ambos) proporcionan un servicio así.

R.9. Recuerde que TCP puede mejorarse con SSL para proporcionar servicios de seguridad proceso a proceso, incluyendo mecanismos de cifrado. ¿En qué capa opera SSL, en la capa de transporte o en la capa de aplicación? Si el desarrollador de la aplicación desea mejorar TCP con SSL, ¿qué tendrá que hacer?

SECCIONES 2.2–2.5

R.10. ¿Qué quiere decir el término protocolo de acuerdo?

R.11. ¿Por qué HTTP, FTP, SMTP y POP3 se ejecutan sobre TCP en lugar de sobre UDP?

R.12. Un sitio de comercio electrónico desea mantener un registro de compras para cada uno de sus clientes. Describa cómo se puede hacer esto utilizando cookies.

R.13. Describa cómo el almacenamiento en caché web puede reducir el retardo de recepción de un objeto solicitado. ¿Reducirá este tipo de almacenamiento el retardo de todos los objetos solicitados por el usuario o sólo el de algunos objetos? ¿Por qué?

R.14. Establezca una sesión Telnet en un servidor web y envíe un mensaje de solicitud de varias líneas. Incluya en dicho mensaje la línea de cabecera `If-modified-since:` para forzar un mensaje de respuesta con el código de estado `304 Not Modified`.

R.15. ¿Por qué se dice que FTP envía la información de control "fuera de banda"?

R.16. Suponga que Alicia, que dispone de una cuenta de correo electrónico web (como por ejemplo Hotmail o gmail), envía un mensaje a Benito, que accede a su correo almacenado en su servidor de correo utilizando POP3. Explique cómo se transmite el mensaje desde el host de Alicia hasta el de Benito. Asegúrese de citar la serie de protocolos de la capa de aplicación que se utilizan para llevar el mensaje de un host al otro.

R.17. Imprima la cabecera de un mensaje de correo electrónico que haya recibido recientemente. ¿Cuántas líneas de cabecera `Received:` contiene? Analice cada una de las líneas de cabecera del mensaje.

R.18. Desde la perspectiva de un usuario, ¿cuál es la diferencia entre el modo “descargar y borrar” y el modo “descargar y mantener” en POP3?

R.19. ¿Pueden el servidor web y el servidor de correo electrónico de una organización tener exactamente el mismo alias para un nombre de host (por ejemplo, `foo.com`)? ¿Cuál sería el tipo especificado en el registro de recurso (RR) que contiene el nombre de host del servidor de correo?

SECCIÓN 2.6

R.20. En BitTorrent, suponga que Alicia proporciona fragmentos a Benito a intervalos de 30 segundos. ¿Devolverá necesariamente Benito el favor y proporcionará fragmentos a Alicia en el mismo intervalo de tiempo? ¿Por qué?

R.21. Suponga que un nuevo par Alicia se une a BitTorrent sin tener en su posesión ningún fragmento. Dado que no posee fragmentos, no puede convertirse en uno de los cuatro principales suministradores de ninguno de los otros pares, ya que no tiene nada que suministrar. ¿Cómo obtendrá entonces Alicia su primer fragmento?

R.22. ¿Qué es una red solapada? ¿Contiene routers? ¿Cuáles son las fronteras en una red solapada? ¿Cómo se crea y se mantiene la red solapada que se encarga de distribuir las consultas?

R.23. ¿En qué sentido la mensajería instantánea con un índice centralizado es un híbrido de las arquitecturas cliente-servidor y P2P?

R.24. Considere una DHT con una topología de red solapada en malla (es decir, cada par controla a todos los demás pares del sistema). ¿Cuáles son las ventajas y desventajas de un diseño de este tipo? ¿Cuáles son las ventajas y desventajas de una DHT circular (sin atajos)?

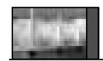
R.25. Skype utiliza técnicas P2P para dos funciones importantes. ¿Cuáles son dichas funciones?

R.26. Cite al menos cuatro aplicaciones distintas que se adapten de forma natural a las arquitecturas P2P. (*Sugerencia:* la distribución de archivos y la mensajería instantánea son dos de ellas.)

SECCIONES 2.7–2.8

R.27. El servidor UDP descrito en la Sección 2.8 sólo necesitaba un socket, mientras que el servidor TCP descrito en la Sección 2.7 necesitaba dos. ¿Por qué? Si el servidor TCP tuviera que soportar n conexiones simultáneas, cada una procedente de un host cliente distinto, ¿cuántos sockets necesitaría el servidor TCP?

R.28. En la aplicación cliente-servidor sobre TCP descrita en la Sección 2.7, ¿por qué tiene que ser ejecutado el programa servidor antes que el programa cliente? En la aplicación cliente-servidor sobre UDP descrita en la Sección 2.8, ¿por qué el programa cliente puede ejecutarse antes que el programa servidor?



Problemas

P1. ¿Verdadero o falso?

a. Un usuario solicita una página web que consta de texto y tres imágenes. Para obtener esa página, el cliente envía un mensaje de solicitud y recibe cuatro mensajes de respuesta.

b. Dos páginas web diferentes (por ejemplo, `www.mit.edu/research.html` y `www.mit.edu/students.html`) se pueden enviar a través de la misma conexión persistente.

c. Con las conexiones no persistentes entre un navegador y un servidor de origen, un único segmento TCP puede transportar dos mensajes de solicitud HTTP distintos.

d. La línea de cabecera Date: del mensaje de respuesta HTTP indica cuándo el objeto fue modificado por última vez.

e. Los mensajes de respuesta HTTP nunca incluyen un cuerpo de mensaje vacío.

P2. Lea el documento RFC 959 relativo a FTP. Enumere todos los comandos cliente que están soportados por dicho documento.

P3. Un cliente HTTP desea recuperar un documento web que se encuentra en un URL dado. Inicialmente, la dirección IP del servidor HTTP es desconocida. ¿Qué protocolos de la capa de aplicación y de la capa de transporte además de HTTP son necesarios en este escenario?

P4. La siguiente cadena de caracteres ASCII ha sido capturada por Wireshark cuando el navegador enviaba un mensaje GET HTTP (es decir, éste es el contenido real de un mensaje GET HTTP). Los caracteres *<cr><lf>* representan el retorno de carro y el salto de línea (es decir, la cadena de caracteres en cursiva *<cr>* del texto que sigue a este párrafo representa el carácter de retorno de carro contenido en dicho punto de la cabecera HTTP). Responda a las siguientes cuestiones, indicando en qué parte del siguiente mensaje GET HTTP se encuentra la respuesta.

```
GET /cs453/index.html HTTP/1.1<cr><lf>Host: gai
a.cs.umass.edu<cr><lf>User-Agent: Mozilla/5.0 (
Windows;U; Windows NT 5.1; en-US; rv:1.7.2) Gec
ko/20040804 Netscape/7.2 (ax) <cr><lf>Accept:ex
t/xml, application/xml, application/xhtml+xml, text
/html;q=0.9, text/plain;q=0.8,image/png,*/*;q=0.5
<cr><lf>Accept-Language: en-us,en;q=0.5<cr><lf>Accept-
Encoding: zip,deflate<cr><lf>Accept-Charset: ISO
-8859-1,utf-8;q=0.7,*;q=0.7<cr><lf>Keep-Alive: 300<cr>
<lf>Connection:keep-alive<cr><lf><cr><lf>
```

a. ¿Cuál es el URL del documento solicitado por el navegador?

b. ¿Qué versión de HTTP se está ejecutando en el navegador?

c. ¿Qué tipo de conexión solicita el navegador, persistente o no persistente?

d. ¿Cuál es la dirección IP del host en el que se está ejecutando el navegador?

e. ¿Qué tipo de navegador inicia este mensaje? ¿Por qué es necesario indicar el tipo de navegador en un mensaje de solicitud HTTP?

P5. El siguiente texto muestra la respuesta devuelta por el servidor al mensaje de solicitud GET HTTP del problema anterior. Responda a las siguientes cuestiones, indicando en qué parte del siguiente mensaje se encuentran las respuestas.

```
HTTP/1.1 200 OK<cr><lf>Date: Tue, 07 Mar 2008
12:39:45GMT<cr><lf>Server: Apache/2.0.52 (Fedora)
<cr><lf>Last-Modified: Sat, 10 Dec2005 18:27:46
GMT<cr><lf>ETag: "526c3-f22-a88a4c80"<cr><lf>
Accept-Ranges: bytes<cr><lf>Content-Length: 3874<cr><lf>
Keep-Alive: timeout=max=100<cr><lf>Connection:
Keep-Alive<cr><lf>Content-Type: text/html; charset=
ISO-8859-1<cr><lf><cr><lf><!doctype html public "-
//w3c//dtd html 4.0 transitional//en"><lf><html><lf>
<head><lf> <meta http-equiv="Content-Type"
content="text/html; charset=iso-8859-1"><lf>
<meta name="GENERATOR" content="Mozilla/4.79 [en] (Windows NT
5.0; U) Netscape]"><lf> <title>CMPSCI 453 / 591 /
NTU-ST550A Spring 2005 homepage</title><lf></head><lf>
<aquí continúa el texto del documento (no mostrado)>
```

a. ¿Ha podido el servidor encontrar el documento? ¿En qué momento se suministró la respuesta con el documento?

b. ¿Cuándo fue modificado por última vez el documento?

c. ¿Cuántos bytes contiene el documento devuelto?

d. ¿Cuáles son los primeros cinco bytes del documento que se está devolviendo? ¿Ha acordado el servidor emplear una conexión persistente?

P6. Utilice la especificación HTTP/1.1 (RFC 2616) para responder a las siguientes cuestiones:

a. Explique el mecanismo de señalización entre el cliente y el servidor para indicar que se está cerrando una conexión persistente. ¿Quién puede señalizar el cierre de la conexión, el cliente, el servidor o ambos?

b. ¿Qué servicios de cifrado proporciona HTTP?

c. ¿Puede un cliente abrir tres o más conexiones simultáneas con un determinado servidor?

d. Un servidor o un cliente pueden cerrar una conexión de transporte entre ellos si uno detecta que la conexión ha estado inactiva durante un cierto tiempo. ¿Es posible que un lado inicie el cierre de una conexión mientras que el otro lado está transmitiendo datos a través de dicha conexión? Explique su respuesta.

P7. Suponga que en su navegador hace clic en un vínculo a una página web. La dirección IP correspondiente al URL asociado no está almacenado en la caché de su host local, por lo que es necesario realizar una búsqueda DNS para obtener la dirección IP. Suponga que antes de que su host reciba la dirección IP de DNS se han visitado n servidores DNS y que los tiempos de ida y vuelta (RTT) de las sucesivas visitas son

$RTT_1, \ldots, RTT_n$. Suponga además que la página web asociada con el vínculo contiene exactamente un objeto, que consta de un pequeño fragmento de texto HTML. Sea RTT_0 el tiempo RTT entre el host local y el servidor que contiene el objeto. Suponiendo un tiempo de transmisión despreciable para el objeto, ¿cuánto tiempo transcurre desde que el cliente hace clic en el vínculo hasta que recibe el objeto?

P8. Continuando con el Problema P7, suponga que el archivo HTML hace referencia a ocho objetos muy pequeños que se encuentran en el mismo servidor. Despreciando los tiempos de transmisión, ¿cuánto tiempo transcurre si se utiliza

a. HTTP no persistente sin conexiones TCP en paralelo?

b. HTTP no persistente con el navegador configurado para 5 conexiones paralelo?

c. HTTP persistente?

P9. En la red institucional conectada a Internet de la Figura 2.12, suponga que el tamaño medio de objeto es de 850.000 bits y que la tasa media de solicitudes de los navegadores de la institución a los servidores de origen es de 16 solicitudes por segundo. Suponga también que el tiempo que se tarda desde que el router en el lado de Internet del enlace de acceso reenvía una solicitud HTTP hasta que recibe la respuesta es, como media, de tres segundos (véase la Sección 2.2.5). Modele el tiempo medio de respuesta total como la suma del retardo medio de acceso (es decir, el retardo desde el router de Internet al router de la institución) y el retardo medio de Internet. Para el retardo medio de acceso, utilice la expresión /(1 –), donde es el tiempo medio requerido para enviar un objeto a través del enlace de acceso y es la tasa de llegada de los objetos al enlace de acceso.

a. Calcule el tiempo medio de respuesta total.

b. Ahora suponga que hay instalada una caché en la LAN institucional. Suponga que la tasa de fallos es de 0,4. Calcule el tiempo de respuesta total.

P10. Dispone de un enlace corto de 10 metros a través del cual un emisor puede transmitir a una velocidad de 150 bits/segundo en ambos sentidos. Suponga que los paquetes de datos tienen una longitud de 100.000 bits y los paquetes que contienen sólo comandos de control (por ejemplo, ACK o de acuerdo) tienen una longitud de 200 bits. Suponga que hay N conexiones en paralelo y que cada una utiliza $1/N$ del ancho de banda del enlace. Considere ahora el protocolo HTTP y suponga que cada objeto descargado es de 100 kbytes de largo y que el objeto inicialmente descargado contiene 10 objetos referenciados procedentes del mismo emisor. ¿Tiene sentido en este caso realizar descargas en paralelo mediante instancias paralelas de HTTP no persistente? Considere ahora HTTP persistente. ¿Cabe esperar alguna ventaja significativa respecto del caso no persistente? Justifique y explique su respuesta.

P11. Continuando con el escenario del problema anterior, suponga que Benito comparte el enlace con otros cuatro usuarios. Benito utiliza instancias paralelas de HTTP no persistente y los otros cuatro usuarios utilizan HTTP no persistente sin descargas en paralelo.

a. ¿Le ayudan a Benito las conexiones en paralelo a obtener las páginas más rápidamente? ¿Por qué?

b. Si los cinco usuarios abren cinco instancias paralelas de HTTP no persistente, ¿seguirán siendo beneficiosas las conexiones en paralelo de Benito? ¿Por qué?

P12. Escriba un programa TCP simple para un servidor que acepte líneas de entrada procedentes de un cliente y muestre dichas líneas en la salida estándar del servidor. (Puede realizar esta tarea modificando el programa `TCPServidor.java` visto en el capítulo.) Compile y ejecute su programa. En cualquier otra máquina que disponga de un navegador web, configure el servidor proxy en el navegador para que apunte al host que está ejecutando su programa servidor; configure también el número de puerto de la forma apropiada. Su navegador deberá ahora enviar sus mensajes de solicitud GET a su servidor y el servidor tendrá que mostrar dichos mensajes en su salida estándar. Utilice esta plataforma para determinar si su navegador genera mensajes GET condicionales para los objetos almacenados localmente en la caché.

P13. ¿Cuál es la diferencia entre `MAIL FROM`: en SMTP y `From`: en el propio mensaje de correo?

P14. ¿Cómo marca SMTP el final del cuerpo de un mensaje? ¿Cómo lo hace HTTP? ¿Puede HTTP utilizar el mismo método que SMTP para marcar el final del cuerpo de un mensaje? Explique su respuesta.

P15. Lea el documento RFC 5321 dedicado a SMTP. ¿Qué quiere decir MTA? Considere el siguiente mensaje de correo basura recibido (modificado a partir de un correo basura real). Suponiendo que únicamente el remitente de este mensaje de correo es malicioso y que los demás hosts son honestos, identifique al host malicioso que ha generado este correo basura.

```
From - Fri Nov 07 13:41:30 2008
Return-Path: <tennis5@pp33head.com>
Received: from barmail.cs.umass.edu
(barmail.cs.umass.edu [128.119.240.3]) by cs.umass.edu
(8.13.1/8.12.6) for <hg@cs.umass.edu>; Fri, 7 Nov 2008
13:27:10 -0500
Received: from asusus-4b96 (localhost [127.0.0.1]) by
barmail.cs.umass.edu (Spam Firewall) for
<hg@cs.umass.edu>; Fri,  7 Nov 2008 13:27:07 -0500
(EST)
Received: from asusus-4b96 ([58.88.21.177]) by
barmail.cs.umass.edu for <hg@cs.umass.edu>; Fri,
07 Nov 2008 13:27:07 -0500 (EST)
Received: from [58.88.21.177] by
inbnd55.exchangeddd.com; Sat, 8 Nov 2008 01:27:07 +0700
From: "Jonny" <tennis5@pp33head.com>
To: <hg@cs.umass.edu>
Subject: Cómo asegurar sus ahorros
```

P16. Lea el documento RFC 1939 dedicado a POP3. ¿Cuál es el propósito del comando UIDL POP3?

P17. Imagine que accede a su correo electrónico utilizando POP3.

a. Suponga que ha configurado su cliente de correo POP para operar en el modo descargar y borrar. Complete la siguiente transacción:

```
C: list
S: 1 498
S: 2 912
S: .
C: retr 1
S: bla bla ...
S: ..........blah
S: .
?
?
```

b. Suponga que ha configurado su cliente de correo POP para operar en el modo descargar y guardar. Complete la siguiente transacción:

```
C: list
S: 1 498
S: 2 912
S: .
C: retr 1
S: bla bla ...
S: ..........bla
S: .
?
?
```

c. Suponga que ha configurado su cliente de correo POP para operar en el modo descargar y guardar. Utilizando su transcripción del apartado (b), suponga que recupera los mensajes 1 y 2, sale de POP, y cinco minutos más tarde vuelve a acceder otra vez a POP para recuperar un nuevo mensaje de correo. Suponga que en ese intervalo de cinco minutos nadie le ha enviado un nuevo mensaje de correo. Proporcione una transcripción de esta segunda sesión de POP.

P18. a. ¿Qué es una base de datos *whois*?

b. Utilice varias bases de datos *whois* de Internet para obtener los nombres de dos servidores DNS. Indique qué bases de datos *whois* ha utilizado.

c. Utilice el programa nslookup en su host local para enviar consultas DNS a tres servidores DNS: su servidor DNS local y los dos servidores DNS que haya encontrado en el apartado (b). Intente realizar consultas para obtener registros de recursos de tipo A, NS y MX. Escriba un resumen de sus hallazgos.

d. Utilice el programa nslookup para localizar un servidor web que tenga varias direcciones IP. ¿Tiene el servidor web de su institución (centro de estudios o empresa) varias direcciones IP?

e. Utilice la base de datos *whois* ARIN para determinar el rango de direcciones IP utilizado en su universidad.

f. Describa cómo un atacante puede utilizar las bases de datos *whois* y la herramienta nslookup para realizar labores de reconocimiento en una institución antes de lanzar un ataque.

g. Explique por qué las bases de datos *whois* deben estar disponibles públicamente.

P19. En este problema empleará la útil herramienta *dig*, disponible en los hosts Unix y Linux para explorar la jerarquía de los servidores DNS. Como se muestra en la Figura 2.21, un servidor DNS que se encuentra en un nivel superior de la jerarquía DNS delega una consulta DNS en un servidor DNS que se encuentra en un nivel más bajo de la jerarquía, devolviendo al cliente DNS el nombre de dicho servidor DNS del nivel más bajo. Lea primero la página de manual dedicada a *dig* y, a continuación, responda a las siguientes preguntas.

a. Comenzando por un servidor DNS raíz (uno de los servidores raíz [a-m].root-servers.net), inicie una secuencia de consultas para obtener la dirección IP del servidor web de su departamento, utilizando la herramienta *dig*. Visualice la lista de nombres de los servidores DNS incluidos en la cadena de delegación que ha obtenido como respuesta a su consulta.

b. Repita el apartado (a) para varios sitios web populares, como por ejemplo google.com, yahoo.com o amazon.com.

P20. Suponga que puede acceder a las cachés de los servidores DNS locales de su departamento. ¿Puede proponer una forma de determinar de manera aproximada los servidores web (situados fuera de su departamento) que son más populares entre los usuarios del departamento? Explique su respuesta.

P21. Suponga que su departamento tiene un servidor DNS local para todas las computadoras del departamento. Usted es un usuario normal (es decir, no es un administrador de la red o del sistema). ¿Puede determinar de alguna manera si desde alguna computadora de su departamento se ha accedido hace unos pocos segundos a un determinado sitio web externo? Explique su respuesta.

P22. Desea distribuir un archivo de F = 15 Gbits a N pares. El servidor tiene una velocidad de carga de u_s = 30 Mbps, y cada par tiene una velocidad de descarga de d_i = 2 Mbps y una velocidad de carga igual a u. Para N = 10, 100 y 1.000, y u = 300 kbps, 700 kbps y 2 Mbps, prepare una gráfica que proporcione el tiempo mínimo de distribución para cada una de las combinaciones de N y u, tanto para una distribución cliente-servidor como para una distribución P2P.

P23. Desea distribuir un archivo de F bits a N pares utilizando una arquitectura cliente-servidor. Suponga un modelo flexible, en el que el servidor puede transmitir simultáneamente a varios pares, transmitiendo a cada par a distintas velocidades, siempre y cuando la velocidad combinada no sea mayor que u_s.

a. Suponga que $u_s/N \quad d_{mín}$. Especifique un esquema de distribución que tenga un tiempo de distribución de NF/u_s.

b. Suponga que $u_s/N \quad d_{mín}$. Especifique un esquema de distribución que tenga un tiempo de distribución de $F/d_{mín}$.

c. Demuestre que, en general, el tiempo mínimo de distribución está dado por $máx\{NF/u_s, F/d_{mín}\}$.

P24. Desea distribuir un archivo de F bits a N pares utilizando una arquitectura P2P. Suponga un modelo flexible. Con el fin de simplificar, suponga que $d_{mín}$ es muy grande, por lo que el ancho de banda de descarga de los pares no es nunca un cuello de botella.

a. Suponga que $u_s \quad (u_s + u_1 + \ldots + u_N)/N$. Especifique un esquema de distribución que tenga un tiempo de distribución de F/u_s.

b. Suponga que $u_s \quad (u_s + u_1 + \ldots + u_N)/N$. Especifique un esquema de distribución que tenga un tiempo de distribución de $NF/(u_s + u_1 + \ldots + u_N)$.

c. Demuestre que, en general, el tiempo mínimo de distribución está dado por máx$\{F/u_s, NF/(u_s + u_1 + \ldots + u_N)\}$.

P25. Considere una red solapada con N pares activos, disponiendo cada pareja de pares de una conexión TCP activa. Suponga también que las conexiones TCP atraviesan un total de M routers. ¿Cuántos nodos y fronteras existen en la correspondiente red solapada?

P26. Suponga que Benito se une a un torrente BitTorrent, pero no desea suministrar datos a otros pares (lo que se denomina “ir por libre”).

a. Benito afirma que puede recibir una copia completa del archivo compartido por el conjunto de usuarios. ¿Es correcto lo que dice Benito? ¿Por qué?

b. Benito añade que puede hacer más eficientes sus descargas utilizando varias computadoras (con distintas direcciones IP) del laboratorio de su departamento. ¿Cómo puede hacer esto?

P27. En este problema, el objetivo es determinar la eficiencia de un sistema de compartición de archivos P2P como BitTorrent. Considere los pares Benito y Alicia. Éstos se unen a un torrente en el que, en total, hay M pares (incluyendo a Benito y Alicia) que están compartiendo un archivo que consta de N fragmentos. Suponga que en un instante determinado t, los fragmentos que tiene un par están seleccionados de forma aleatoria y uniforme de entre los N fragmentos y que ningún par tiene todos los N fragmentos. Responda a las siguientes preguntas.

a. ¿Cuál es la probabilidad de que Benito tenga todos los fragmentos que tiene Alicia, si expresamos el número de fragmentos que tiene cada uno como n_b (Benito) y n_a (Alicia)?

b. Elimine algunas de las suposiciones del apartado (a) para calcular la probabilidad de que Benito tenga todos los fragmentos que tiene Alicia, si ésta tiene n_a fragmentos?

c. Suponga que cada par de BitTorrent tiene cinco vecinos. ¿Cuál es la probabilidad de que Benito tenga datos que sean del interés de al menos uno de sus cinco vecinos?

P28. En el ejemplo de la DHT circular de la Sección 2.6.2, suponga que el par 3 sabe que el par 5 ha abandonado la red. ¿Cómo actualiza el par 3 la información de estado de su sucesor? ¿Qué par será ahora su primer sucesor? ¿Y su segundo sucesor?

P29. En el ejemplo de la DHT circular de la Sección 2.6.2, suponga que un nuevo par 6 desea unirse a la DHT y que inicialmente el par 6 sólo conoce la dirección IP del par 15. ¿Qué pasos tendrá que dar?

P30. Sea una red DHT circular con nodos e identificadores de clave en el rango de [0, 63]. Suponga que hay ocho pares cuyos identificadores son 0, 8, 16, 24, 32, 40, 48 y 56.

a. Suponga que cada par puede tener un par de atajo. Para cada uno de los ocho pares, determine su par de atajo, de manera que el número de mensajes enviados para cualquier consulta (iniciada en cualquier par) sea mínimo.

b. Repita el apartado (a) pero ahora permita que cada par tenga dos pares de atajo.

P31. Puesto que un número entero perteneciente al intervalo [0, 2^n 1] se puede expresar como un número binario de n bits en una DHT, cada clave se puede expresar como $k = (k_0, k_1, \ldots, k_{n-1})$ y cada identificador de un par como $p = (p_0, p_1, \ldots, p_{n-1})$, definimos la distancia XOR entre una clave k y un par p como

$$d(k, p) = \sum_{j=0}^{n-1} | k_j - p_j | 2^j$$

Explique cómo se puede utilizar esta métrica para asignar parejas (clave, valor) a los pares. (Para obtener más información sobre cómo crear una DHT eficiente utilizando esta métrica natural, consulte [Maymounkov 2002] donde se describe la DHT Kademlia.)

P32. Considere una versión generalizada del esquema descrito en el problema anterior. En lugar de utilizar números binarios, ahora vamos a tratar las claves y los identificadores de los pares como números en base b, donde $b > 2$, y luego utilizaremos la métrica del problema anterior para diseñar una DHT (sustituyendo 2 por b). Compare esta DHT basada en números en base b con la DHT basada en números binarios. En el caso peor, ¿qué DHT genera más mensajes por consulta? ¿Por qué?

P33. Dado que las DHT son redes solapadas, no necesariamente se corresponden con las redes físicas subyacentes en el sentido de que dos pares vecinos pueden estar físicamente muy alejados; por ejemplo, un par podría encontrarse en Asia y su vecino estar en América del Norte. Si asignamos identificadores de forma aleatoria y uniforme a los pares que acaban de unirse a la red, ¿podría este esquema de asignaciones hacer que se produjera esa no correspondencia? Explique su respuesta. ¿Cómo podría esa no correspondencia afectar al rendimiento de la DHT?

P34. Instale y compile los programas Java `TCPCliente` y `UDPCliente` en un host y los programas `TCPServidor` y `UDPServidor` en otro host.

a. Suponga que ejecuta `TCPCliente` antes que el programa `TCPServidor`. ¿Qué ocurrirá? ¿Por qué?

b. Suponga que ejecuta el programa `UDPCliente` antes que `UDPServidor`. ¿Qué ocurrirá? ¿Por qué?

c. ¿Qué ocurrirá si utiliza diferentes números de puerto para los lados de cliente y de servidor?

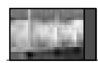
P35. Suponga que en el programa `UDPClient.java` sustituimos la línea

```
DatagramSocket socketCliente= new DatagramSocket();
```

por

```
DatagramSocket socketCliente= new DatagramSocket(5432);
```

¿Será necesario modificar el programa `UDPServidor.java`? ¿Cuáles son los números de puerto para los sockets en `UDPClient` y `UDPServidor`? ¿Cuáles eran antes de realizar este cambio?

 Preguntas para la discusión

D1. ¿Por qué cree que las aplicaciones de compartición de archivos P2P son tan populares? ¿Es acaso porque distribuyen música y vídeos de foma gratuita (algo que puede ser ilegal)? ¿Es porque su enorme cantidad de servidores responden de forma eficiente a una enorme demanda de megabytes? ¿O es por todas estas razones?

D2. Lea el documento "The Darknet and the Future of Content Distribution" de Biddle, England, Peinado y Willman [Biddle 2003]. ¿Está de acuerdo con los puntos de vista de los autores? ¿Por qué?

D3. Los sitios web dedicados al comercio electrónico y otros suelen disponer de bases de datos back-end. ¿Cómo se comunican los servidores HTTP con estas bases de datos back-end?

D4. ¿Cómo se puede configurar un navegador para almacenamiento en caché local? ¿De qué opciones de almacenamiento en caché dispone?

D5. ¿Puede configurar su navegador para abrir varios conexiones simultáneas con un sitio web? ¿Cuáles son las ventajas y desventajas de disponer de un gran número de conexiones TCP simultáneas?

D6. Hemos visto que los sockets TCP de Internet tratan los datos que están siendo enviados como un flujo de bytes, pero los sockets UDP reconocen los límites de los mensajes. Cite una ventaja y una desventaja de las API orientadas a byte y de tener una API que reconozca y mantenga explícitamente los límites de los mensajes definidos por la aplicación?

D7. ¿Qué es un servidor web Apache? ¿Cuánto cuesta? ¿Qué funcionalidad proporciona actualmente?

D8. Muchos clientes BitTorrent utilizan tablas DHT para crear un tracker distribuido. Para dichas DHT, ¿cuál es la "clave" y cuál es el "valor"?

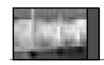
D9. Imagine que las organizaciones de estándares web deciden cambiar el convenio de denominación, de modo que cada objeto se nombre y se referencie mediante un único nombre que sea independiente de la ubicación (lo que se denomina un URN). Indique algunos de los problemas que implicaría tal cambio.

D10. ¿Hay actualmente compañías que distribuyan a través de Internet televisión en directo? En caso afirmativo, ¿utilizan dichas empresas arquitecturas cliente-servidor o P2P?

D11. ¿Están las empresas proporcionando actualmente un servicio de vídeo a la carta a través de Internet utilizando una arquitectura P2P?

D12. ¿Cómo proporciona Skype un servicio PC a teléfono a tantos países de destino distintos?

D13. ¿Cuáles son hoy día algunos de los clientes BitTorrent más populares?



Tareas sobre programación de sockets

Tarea 1: servidor web multihebra

Al terminar esta tarea de programación, habrá desarrollado, en Java, un servidor web multihebra capaz de servir varias solicitudes en paralelo. Va a implementar la versión 1.0 de HTTP, tal y como se define en el documento RFC 1945.

HTTP/1.0 crea una conexión TCP separada para cada par solicitud/respuesta. Una hebra separada gestiona cada una de las conexiones. También habrá una hebra principal, en la que el servidor escuche a los clientes que deseen establecer una conexión. Para simplificar la tarea de programación, desarrollaremos el código en dos etapas. En la primera, escribirá el código para un servidor multihebra que simplemente muestre el contenido del mensaje de solicitud HTTP que haya recibido. Una vez que este programa se ejecute correctamente, deberá añadir el código necesario para generar una respuesta apropiada.

A medida que vaya desarrollando el código, puede probar su servidor con un navegador web. Pero recuerde que no está proporcionando el servicio a través del puerto estándar 80, por lo que tendrá que especificar el número de puerto dentro del URL que proporcione al navegador. Por ejemplo, si el nombre de host es `host.unaEscuela.edu`, su servidor está a la escucha en el puerto 6789 y desea recuperar el archivo `index.html`, entonces tendrá que especificar el siguiente URL en el navegador:

```
http://host.unaescuela.edu:6789/index.html
```

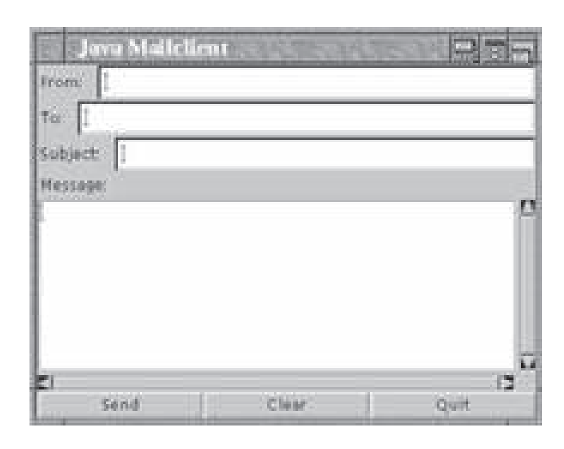
Cuando su servidor detecte un error, deberá enviar un mensaje de respuesta con un código fuente HTML apropiado, de manera que la información acerca del error se muestre en la ventana del navegador. Puede encontrar todos los detalles de esta tarea, así como útiles fragmentos en código Java en el sitio web http://www.awl.com/kurose-ross.

Tarea 2: cliente de correo

En esta tarea tendrá que desarrollar, en Java, un agente de usuario de correo con las siguientes características:

- Debe proporcionar una interfaz gráfica al emisor, con campos para el servidor de correo local, la dirección de correo electrónico del emisor, la dirección de correo electrónico del receptor, el asunto del mensaje y el propio mensaje.
- Debe establecer una conexión TCP entre el cliente de correo y el servidor de correo local, enviar comandos SMTP al servidor de correo local, y recibir y procesar los comandos SMTP procedentes del servidor de correo local.

La interfaz tendrá que ser similar a la captura de pantalla mostrada en la página siguiente. Debe desarrollar el agente de usuario de manera que envíe un mensaje de correo electrónico como máximo a un destinatario cada vez. Además, el agente de usuario supondrá que la parte del dominio de la dirección de correo electrónico del destinatario es el nombre canónico del servidor SMTP del destinatario. (El agente de usuario no realizará una búsqueda DNS para obtener un registro MX, por lo que el emisor debe suministrar el nombre real del servidor de correo.) En el sitio web del libro, http://www.awl.com/kurose-ross, puede encontrar todos los detalles de esta tarea, así como útiles fragmentos de código Java.



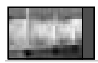
Tarea 3: generador de comandos Ping UDP

En esta práctica de laboratorio tendrá que implementar un servidor y un cliente Ping basados en UDP. La funcionalidad proporcionada por estos programas es similar al programa estándar Ping disponible en los sistemas operativos actuales. El programa estándar Ping envía mensajes ECHO ICMP (*Internet Control Message Protocol*), que la máquina remota devuelve al emisor. El emisor puede así determinar el tiempo de ida y vuelta transcurrido desde que ejecuta el comando Ping hasta que recibe la respuesta de la otra computadora.

Java no proporciona ninguna funcionalidad para enviar ni recibir mensajes ICMP, razón por la que en esta práctica de laboratorio tendrá que implementar un programa Ping en la capa de aplicación con mensajes y sockets UDP estándar. En el sitio web del libro, http://www .awl.com/kurose-ross, puede encontrar todos los detalles de esta tarea, así como útiles fragmentos de código Java.

Tarea 4: servidor proxy web

En esta práctica de laboratorio tendrá que desarrollar un servidor proxy web simple, que también sea capaz de almacenar en caché páginas web. Este servidor aceptará un mensaje GET de un navegador, lo reenviará al servidor web de destino, recibirá el mensaje de respuesta HTTP del servidor de destino y reenviará ese mensaje de respuesta HTTP al navegador. Se trata de un servidor proxy muy sencillo; basta con que comprenda solicitudes GET simples. Sin embargo, el servidor podrá manejar toda clase de objetos, no sólo páginas HTML, incluyendo imágenes. En el sitio web del libro, http://www .awl.com/kurose-ross, puede encontrar todos los detalles de esta tarea, así como útiles fragmentos de código Java.

 Prácticas de laboratorio con Wireshark

Práctica de laboratorio con Wireshark: HTTP

En la práctica de laboratorio 1 nos hemos familiarizado con el husmeador de paquetes (*sniffer*) Wireshark, así que ya estamos preparados para utilizar Wireshark e investigar el funcio-

namiento de los protocolos. En esta práctica de laboratorio exploraremos varios aspectos del protocolo HTTP: la interacción básica GET/respuesta, los formatos de los mensajes HTTP, la recuperación de archivos HTML de gran tamaño, la recuperación de archivos HTML con direcciones URL incrustadas, las conexiones persistentes y no persistentes, y la autenticación y la seguridad de HTTP.

Al igual que todas las prácticas de laboratorio con Wireshark, la descripción completa de esta práctica se encuentra en el sitio web http://www.awl.com/kurose-ross.

Práctica de laboratorio con Wireshark: DNS

En esta práctica de laboratorio echaremos un rápido vistazo al lado del cliente de DNS, el protocolo que traduce los nombres de host Internet en direcciones IP. Como hemos visto en la Sección 2.5, el papel del cliente en el protocolo DNS es relativamente simple: un cliente envía una consulta a su servidor DNS local y recibe una respuesta. Sin embargo, son muchas las cosas que suceden por debajo, invisibles para los clientes DNS, ya que los servidores DNS jerárquicos se comunican entre sí de forma recursiva o iterativa para resolver la consulta DNS del cliente. No obstante, desde el punto de vista del cliente DNS, el protocolo es muy simple: se plantea una consulta al servidor DNS local y dicho servidor devuelve una respuesta. En esta práctica de laboratorio vamos a observar al protocolo DNS en acción.

La descripción completa de esta práctica de laboratorio está disponible en el sitio web del libro, http://www.awl.com/kurose-ross.

Bram Cohen



Bram Cohen es Director Científico y co-fundador de BitTorrent, Inc., así como el creador del protocolo de distribución de archivos P2P BitTorrent. También es el co-fundador de CodeCon y co-autor de Codeville. Antes de la creación de BitTorrent, Bram trabajaba en MojoNation. MojoNation permitía a los usuarios descomponer archivos confidenciales en fragmentos cifrados y distribuir esos fragmentos a otras computadoras en las que se estuviera ejecutando el software de MojoNation. Este concepto fue el que inspiró a Bram el desarrollo de BitTorrent. Antes de MojoNation, Bram ya había estado inmerso en el mundo de las punto com, trabajando para varias empresas de Internet a mediados y finales de la década de 1990. Bram creció en Nueva York, se graduó en la Escuela Superior Stuyvesant y cursó sus estudios en la Universidad de Búfalo.

¿Cómo se le ocurrió la idea de desarrollar BitTorrent?

Tenía bastante experiencia laboral en el campo de las redes (protocolos por encima de TCP/UDP) y la implementación de aplicaciones para comunidades de usuarios me parecía el problema no resuelto más interesante que había planteado en ese momento, así que decidí dedicarme a él.

El cálculo fundamental en el que se basa BitTorrent es de una naturaleza muy trivial: hay una gran cantidad de capacidad distribuida entre todas las computadoras del mundo. Son muchas las personas que también se fijaron en esto; pero llevar a cabo una implementación que pudiera gestionar adecuadamente los aspectos logísticos implicados en este problema ya es algo completamente distinto.

¿Cuáles fueron los aspectos más complicados del desarrollo de BitTorrent?

La parte fundamental era acertar con el diseño global y con los aspectos comunitarios del protocolo. Una vez resuelto ese aspecto conceptual, el llevarlo a la práctica era una "simple cuestión de programación". En términos de implementación, la parte más difícil con mucha diferencia fue implementar un sistema fiable. Cuando se está tratando con pares (peers) que no son de confianza, es necesario asumir que cualquiera de ellos puede hacer cualquier cosa en cualquier momento, y es preciso definir algún tipo de comportamiento para todos los casos límite. Tuve que continuar rescribiendo amplias secciones de BitTorrent cuando lo estaba creando, a medida que surgían nuevos problemas y el diseño global iba clarificándose.

Inicialmente, ¿cómo descubrió la gente BitTorrent?

En general, la gente descubrió BitTorrent al dedicarse a realizar descargas. Había algún tipo de contenido que deseaban y ese contenido sólo estaba disponible utilizando BitTorrent, así que lo descargaban de esa forma. Los editores a menudo decidían utilizar BitTorrent simplemente porque no disponían del ancho de banda necesario para distribuir su contenido de ninguna otra forma.

¿Podría decirnos qué piensa acerca de las demandas legales planteadas por algunas organizaciones contra las personas que utilizan programas de compartición de archivos como BitTorrent, con el fin de distribuir películas y música? ¿Alguna vez le han demandado por desarrollar tecnologías con las que se distribuye ilegalmente material sujeto a derechos de autor?

La infracción de los derechos de autor es ilegal, la tecnología no lo es. Nunca me ha demandado nadie, porque yo no he estado involucrado en ningún tipo de actividad que infrinja los derechos de autor. Si lo que te interesa es desarrollar tecnología, debes limitarte a los aspectos tecnológicos.

¿Piensa que pueden aparecer en un futuro próximo otros sistemas de distribución de archivos que sustituyan a BitTorrent? Por ejemplo, ¿podría Microsoft incluir su propio protocolo propietario de distribución de archivos en una futura versión de algún sistema operativo?

Puede que existan otros protocolos comunes en el futuro, pero los principios fundamentales de cómo compartir datos, determinados en el protocolo BitTorrent, es poco probable que cambien. La forma más probable de que se produzca un cambio fundamental sería si se produjera una modificación significativa de la estructura global de Internet, debida a la variación radical de las relaciones entre algunas de las constantes fundamentales, a medida que se incrementen las velocidades. Pero las predicciones para los próximos años lo único que hacen es reforzar aún más el modelo actual.

Hablando en términos más generales, ¿a dónde cree que se dirige Internet? ¿Cuáles cree que son o que serán los desafíos tecnológicos más importantes? ¿Puede ver alguna nueva "aplicación estrella" en el horizonte?

Internet, y la tecnología informática en general, está haciéndose cada vez más ubicua. El iPod nano puede ser uno de los grandes ganadores, especialmente a medida que vayan bajando los precios. El desafío tecnológico actual más interesante es el de recopilar la mayor cantidad posible de datos de todos los dispositivos conectados y hacer que esos datos estén disponibles de una forma accesible y útil. Por ejemplo, casi todos los dispositivos portátiles podrían incluir un mecanismo GPS, y todos los objetos que poseemos, incluyendo la ropa, los juguetes, los electrodomésticos y los muebles, podrían decirnos dónde están cuando los perdemos y proporcionarnos una información completa de su historial, incluyendo las operaciones de mantenimiento necesarias, la utilidad futura esperada, la detección de malos usos, etc. No sólo podríamos obtener información acerca de nuestras pertenencias, sino también acerca de, por ejemplo, el ciclo de vida general de un producto concreto; esta información podría recopilarse de forma muy precisa y la coordinación con otras personas sería mucho más fácil, lo que nos permitiría ir más allá de esa mejora tan simple, pero tan fundamental, derivada del hecho de que ahora las personas pueden localizarse fácilmente cuando todas ellas disponen de teléfonos móviles.

¿Quién le ha servido de inspiración profesionalmente? ¿De qué manera?

No me viene a la mente ningún caso concreto, pero he seguido muy de cerca la mitología general de las empresas de nueva creación en Silicon Valley.

¿Qué consejos les daría a los estudiantes que se inician en el campo de las redes/Internet?

Lo que les diría es que encuentren un tema que no sea por el momento de demasiada actualidad pero en el que crean que se pueden efectuar desarrollos atractivos y que ellos encuentren personalmente interesantes. Y que comiencen a trabajar en eso. También deben tratar de adquirir experiencia profesional en el campo en el que deseen trabajar. La experiencia del mundo real nos enseña qué es importante en ese mundo y eso es algo que casi nunca se puede apreciar de forma objetiva cuando se miran las cosas desde dentro del mundo académico.



CAPÍTULO 3

La capa de transporte

Entre las capas de aplicación y de red se encuentra la capa de transporte, una pieza fundamental de la arquitectura de red en capas. Desempeña el papel crítico de proporcionar directamente servicios de comunicación a los procesos de aplicación que se ejecutan en hosts diferentes. El método didáctico que vamos a aplicar a lo largo de este capítulo va a consistir en alternar las explicaciones sobre los principios de la capa de transporte con explicaciones acerca de cómo estos principios se implementan en los protocolos existentes; como siempre, haremos un especial hincapié en los protocolos de Internet, en particular en los protocolos de transporte TCP y UDP.

Comenzaremos explicando la relación existente entre las capas de transporte y de red. Para ello, examinaremos la primera función crítica de la capa de transporte: ampliar el servicio de entrega de la capa de red entre dos sistemas terminales a un servicio de entrega entre dos procesos de la capa de aplicación que se ejecutan en los sistemas terminales. Ilustraremos esta función con el protocolo de transporte sin conexión de Internet, UDP.

A continuación, volveremos a los principios y afrontaremos uno de los problemas más importantes de las redes de computadoras: cómo dos entidades pueden comunicarse de forma fiable a través de un medio que puede perder o corromper los datos. A través de una serie de escenarios cada vez más complejos (¡y realistas!), construiremos un conjunto de técnicas que estos protocolos de transporte emplean para resolver este problema. Después, mostraremos cómo esos principios están integrados en TCP, el protocolo de transporte orientado a la conexión de Internet.

Luego pasaremos al segundo problema más importante de las redes: controlar la velocidad de transmisión de las entidades de la capa de transporte con el fin de evitar, o recuperarse de, las congestiones que tienen lugar dentro de la red. Consideraremos las causas y las consecuencias de la congestión, así como las técnicas de control de congestión más comúnmente utilizadas. Una vez que haya entendido los problemas que hay detrás de los mecanismos de control de congestión, estudiaremos el método que aplica TCP para controlar la congestión.

3.1 La capa de transporte y sus servicios

En los dos capítulos anteriores hemos visto por encima cuál es la función de la capa de transporte, así como los servicios que proporciona. Repasemos rápidamente lo que ya sabemos acerca de la capa de transporte.

Un protocolo de la capa de transporte proporciona una **comunicación lógica** entre procesos de aplicación que se ejecutan en hosts diferentes. Por *comunicación lógica* queremos decir que, desde la perspectiva de la aplicación, es como si los hosts que ejecutan los procesos estuvieran conectados directamente; en realidad, los hosts pueden encontrarse en puntos opuestos del planeta, conectados mediante numerosos routers y a través de un amplio rango de tipos de enlace. Los procesos de aplicación utilizan la comunicación lógica proporcionada por la capa de transporte para enviarse mensajes entre sí, sin preocuparse por los detalles de la infraestructura física utilizada para transportar estos mensajes. La Figura 3.1 ilustra el concepto de comunicación lógica.

Como se muestra en la Figura 3.1, los protocolos de la capa de transporte están implementados en los sistemas terminales, pero no en los routers de la red. En el lado emisor, la capa de transporte convierte los mensajes que recibe procedentes de un proceso de aplicación emisor en paquetes de la capa de transporte, conocidos como **segmentos** de la capa de transporte en la terminología de Internet. Muy posiblemente, esto se hace dividiendo los mensajes de la aplicación en fragmentos más pequeños y añadiendo una cabecera de la capa de transporte a cada fragmento, con el fin de crear el segmento de la capa de transporte. A continuación, la capa de transporte pasa el segmento a la capa de red del sistema terminal emisor, donde el segmento se encapsula dentro de un paquete de la capa de red (un datagrama) y se envía al destino. Es importante destacar que los routers de la red sólo actúan sobre los campos correspondientes a la capa de red del datagrama; es decir, no examinan los campos del segmento de la capa de transporte encapsulado en el datagrama. En el lado receptor, la capa de red extrae el segmento de la capa de transporte del datagrama y lo sube a la capa de transporte. A continuación, esta capa procesa el segmento recibido, poniendo los datos del segmento a disposición de la aplicación de recepción.

Para las aplicaciones de red puede haber más de un protocolo de la capa de transporte disponible. Por ejemplo, Internet tiene dos protocolos: TCP y UDP. Cada uno de estos protocolos proporciona un conjunto diferente de servicios para la capa de transporte a la aplicación que lo haya invocado.

3.1.1 Relaciones entre las capas de transporte y de red

Recuerde que la capa de transporte se encuentra justo encima de la capa de red dentro de la pila de protocolos. Mientras que un protocolo de la capa de transporte proporciona una comunicación lógica entre *procesos* que se ejecutan en hosts diferentes, un protocolo de la capa de red proporciona una comunicación lógica entre *hosts*. Esta distinción es sutil, pero importante. Examinemos esta distinción con la ayuda de una analogía.

Considere dos viviendas, una situada en la costa este de Estados Unidos y otra en la costa oeste, y que en cada hogar viven una docena de niños. Los niños de la costa este son primos de los niños de la costa oeste. A todos los niños les gusta escribirse, y cada niño escribe a todos sus primos todas las semanas, enviando una carta a través del servicio postal ordinario para cada uno de ellos y empleando un sobre para cada uno. Así, cada casa envía 144 cartas a la otra casa cada semana (estos niños podrían ahorrar mucho dinero si utilizaran

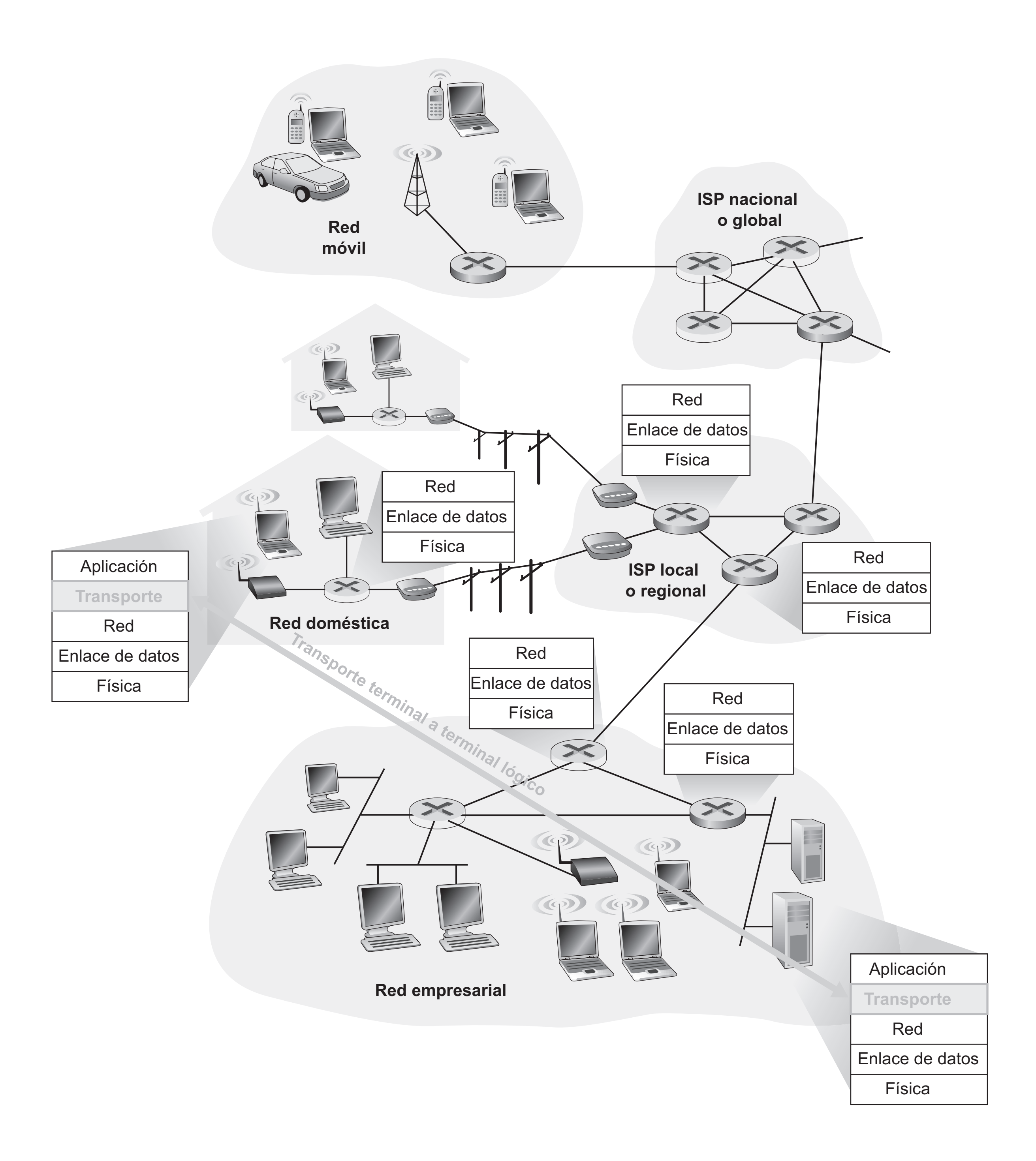


Figura 3.1 • La capa de transporte proporciona una comunicación lógica en lugar de física entre los procesos de aplicación.

el correo electrónico). En cada uno de los hogares, hay un niño (Ann en la costa oeste y Bill en la costa este) responsable de recoger y distribuir el correo. Todas las semanas, Ann visita a sus hermanos y hermanas, les recoge el correo y lo entrega a la persona del servicio postal que pasa a diario por su casa. Cuando las cartas llegan al hogar de la costa oeste, Ann también se ocupa de distribuir el correo al resto de sus hermanos y hermanas. Bill hace el mismo trabajo que Ann en la costa este.

En este ejemplo, el servicio postal proporciona una comunicación lógica entre las dos casas (el servicio postal lleva el correo de una casa a la otra, no de una persona a otra). Por otro lado, Ann y Bill proporcionan una comunicación lógica entre los primos (recogen el correo y se lo entregan a sus hermanos). Observe que desde la perspectiva de los primos, Ann y Bill *son* el servicio de correo, aun siendo ambos solamente una parte (el sistema terminal) del proceso de entrega terminal a terminal. Este ejemplo doméstico es una sencilla analogía que nos permite explicar cómo se relaciona la capa de transporte con la capa de red:

mensajes de la aplicación = las cartas introducidas en los sobres

procesos = los primos

hosts (también denominados sistemas terminales) = las casas

protocolo de la capa de transporte = Ann y Bill

protocolo de la capa de red = el servicio postal (incluyendo a los carteros)

Continuando con esta analogía, fíjese en que Ann y Bill hacen su trabajo dentro de sus respectivas casas; es decir, no están implicados, por ejemplo, en el proceso de ordenación del correo en una oficina de correos intermedia, ni tampoco trasladan el correo de una oficina a otra. De manera similar, los protocolos de la capa de transporte residen en los sistemas terminales. Dentro de un sistema terminal, el protocolo de transporte lleva los mensajes desde los procesos de la aplicación a la frontera de la red (es decir, a la capa de red) y viceversa, pero no tiene nada que ver con cómo se transmiten los mensajes dentro del núcleo de la red. De hecho, como se ilustra en la Figura 3.1, los routers intermedios ni actúan sobre la información que la capa de transporte pueda añadir a los mensajes de la aplicación ni tampoco la reconocen.

Suponga ahora que Ann y Bill se van de vacaciones y que otra pareja de primos, por ejemplo, Susan y Harvey, les sustituyen y son los que recogen y reparten el correo en sus respectivas casas. Lamentablemente para las dos familias, Susan y Harvey no recogen y reparten el correo de la misma forma que lo hacían Ann y Bill. Al ser más pequeños, Susan y Harvey recogen y entregan el correo menos frecuentemente y, ocasionalmente, pierden algunas cartas (que a veces se come el perro). Por tanto, la pareja de primos Susan y Harvey no proporcionan el mismo conjunto de servicios (es decir, el mismo modelo de servicio) que Ann y Bill. De forma análoga, una red de computadoras puede emplear distintos protocolos de transporte, ofreciendo cada uno de ellos un modelo de servicio distinto a las aplicaciones.

Los posibles servicios que Ann y Bill pueden proporcionar están evidentemente restringidos por los posibles servicios que el servicio postal suministra. Por ejemplo, si el servicio postal no especifica el tiempo máximo que se puede tardar en entregar el correo entre ambos hogares (por ejemplo, tres días), entonces Ann y Bill no tienen ninguna forma de garantizar un retardo máximo en la entrega del correo entre cualquier pareja de primos. Del mismo modo, los servicios que un protocolo de transporte puede proporcionar a menudo están restringidos por el modelo de servicio del protocolo de la capa de red subyacente. Si el protocolo de la capa de red no proporciona garantías acerca del retardo ni del ancho de banda para los segmentos de la capa de transporte enviados entre hosts, entonces el protocolo de la capa

de transporte no puede proporcionar ninguna garantía acerca del retardo o del ancho de banda a los mensajes de aplicación enviados entre procesos.

No obstante, el protocolo de transporte *puede* ofrecer ciertos servicios incluso cuando el protocolo de red subyacente no ofrezca el servicio correspondiente en la capa de red. Por ejemplo, como veremos más adelante en el capítulo, un protocolo de transporte puede ofrecer un servicio de transferencia de datos fiable a una aplicación, incluso si el protocolo de red subyacente no es fiable; es decir, incluso si el protocolo de red pierde, altera o duplica paquetes. Otro ejemplo, que examinaremos en el Capítulo 8 cuando hablemos de la seguridad de la red, es que un protocolo de transporte puede emplear mecanismos de cifrado para garantizar que los mensajes de una aplicación no sean leídos por intrusos, incluso aunque la capa de red no pueda garantizar la privacidad de los segmentos de la capa de transporte.

3.1.2 La capa de transporte en Internet

Recuerde que Internet y, de forma más general, las redes TCP/IP, ponen a disposición de la capa de aplicación dos protocolos de la capa de transporte diferentes. Uno de estos protocolos es el Protocolo de datagramas de usuario (**UDP**, *User Datagram Protocol*), que proporciona un servicio sin conexión no fiable a la aplicación que le invoca. El segundo de estos protocolos es el Protocolo de control de transmisión (**TCP**, *Transmission Control Protocol*), que proporciona a la aplicación que le invoca un servicio orientado a la conexión fiable. Al diseñar una aplicación de red, el desarrollador tiene que especificar el uso de uno de estos dos protocolos de transporte. Como hemos visto en las Secciones 2.7 y 2.8, el desarrollador de la aplicación elige entre UDP y TCP cuando crea los sockets.

Para simplificar la terminología, en el contexto de Internet nos referiremos a los paquetes de la capa de transporte como *segmentos*. No obstante, tenemos que decir que, en textos dedicados a Internet, como por ejemplo los RFC, también se emplea el término segmento para hacer referencia a los paquetes de la capa de transporte en el caso de TCP, pero a menudo a los paquetes de UDP se les denomina datagrama. Pero resulta que estos mismos textos dedicados a Internet también utilizan el término *datagrama* para referirse a los paquetes de la capa de red. Por tanto, pensamos que en un libro de introducción a las redes de computadoras como éste, resultará menos confuso hablar de segmentos tanto para referirse a los paquetes TCP como UDP, y reservar el término *datagrama* para los paquetes de la capa de red.

Antes de continuar con esta breve introducción a los protocolos UDP y TCP, nos será útil comentar algunas cosas acerca de la capa de red de Internet (en el Capítulo 4 examinaremos en detalle la capa de red). El protocolo de la capa de red de Internet es el protocolo IP (*Internet Protocol*). IP proporciona una comunicación lógica entre hosts. El modelo de servicio de IP es un **servicio de entrega de mejor esfuerzo (*best effort*)**. Esto quiere decir que IP *hace todo lo que puede* por entregar los segmentos entre los hosts que se están comunicando, *pero no garantiza la entrega.* En particular, no garantiza la entrega de los segmentos, no garantiza que los segmentos se entreguen en orden y no garantiza la integridad de los datos contenidos en los segmentos. Por estas razones, se dice que IP es un **servicio no fiable**. Además, sabemos que todos los hosts tienen al menos una dirección de capa de red, que se conoce como dirección IP. En el Capítulo 4 se estudia en detalle el direccionamiento IP, pero por el momento lo único que necesitamos saber es que *todo host tiene una dirección IP asociada*.

Después de haber echado una ojeada al modelo de servicio de IP, haremos un breve resumen de los modelos de servicio proporcionados por UDP y TCP. La responsabilidad

principal de UDP y TCP es ampliar el servicio de entrega de IP entre dos sistemas terminales a un servicio de entrega entre dos procesos que estén ejecutándose en los sistemas terminales. Extender la entrega host a host a una entrega proceso a proceso es lo que se denomina **multiplexación** y **demultiplexación de la capa de transporte**, tema que desarrollaremos en la siguiente sección. UDP y TCP también proporcionan servicios de comprobación de la integridad de los datos al incluir campos de detección de errores en las cabeceras de sus segmentos. Estos dos servicios de la capa de transporte (entrega de datos proceso a proceso y comprobación de errores) son los dos únicos servicios que ofrece UDP. En particular, al igual que IP, UDP es un servicio no fiable, que no garantiza que los datos enviados por un proceso lleguen intactos (¡o que ni siquiera lleguen!) al proceso de destino. En la Sección 3.3 se aborda en detalle el protocolo UDP.

TCP, por el contrario, ofrece a las aplicaciones varios servicios adicionales. El primero y más importante es que proporciona una **transferencia de datos fiable**. Utilizando técnicas de control de flujo, números de secuencia, mensajes de reconocimiento y temporizadores (técnicas que estudiaremos en detalle en este capítulo), TCP garantiza que los datos transmitidos por el proceso emisor sean entregados al proceso receptor, correctamente y en orden. De este modo, TCP convierte el servicio no fiable de IP entre sistemas terminales en un servicio de transporte de datos fiable entre procesos. TCP también proporciona mecanismos de **control de congestión**. El control de congestión no es tanto un servicio proporcionado a la aplicación invocante, cuanto un servicio que se presta a Internet como un todo, un servicio para el bien común. En otras palabras, los mecanismos de control de congestión de TCP evitan que cualquier conexión TCP inunde con una cantidad de tráfico excesiva los enlaces y routers existentes entre los hosts que están comunicándose. TCP se esfuerza en proporcionar a cada conexión que atraviesa un enlace congestionado la misma cuota de ancho de banda del enlace. Esto se consigue regulando la velocidad a la que los lados emisores de las conexiones TCP pueden enviar tráfico a la red. El tráfico UDP, por el contrario, no está regulado. Una aplicación que emplee el protocolo de transporte UDP puede enviar los datos a la velocidad que le parezca, durante todo el tiempo que quiera.

Un protocolo que proporciona una transferencia de datos fiable y mecanismos de control de congestión necesariamente es un protocolo complejo. Vamos a necesitar varias secciones de este capítulo para examinar los principios de la transferencia de datos fiable y el control de congestión, además de otras secciones adicionales dedicadas a cubrir el propio protocolo TCP. Estos temas se tratan en las Secciones 3.4 a 3.8. El método de trabajo que hemos adoptado en este capítulo es el de alternar entre los principios básicos y el protocolo TCP. Por ejemplo, en primer lugar veremos en qué consiste en general una transferencia de datos fiable y luego veremos específicamente cómo TCP proporciona ese servicio de transferencia de datos fiable. De forma similar, primero definiremos de forma general en qué consiste el mecanismo de control de congestión y luego veremos cómo lo hace TCP. Pero antes de entrar en estos temas, veamos qué es la multiplexación y la demultiplexación de la capa de transporte.

3.2 Multiplexación y demultiplexación

En esta sección vamos a estudiar la multiplexación y demultiplexación de la capa de transporte; es decir, la ampliación del servicio de entrega host a host proporcionado por la capa de red a un servicio de entrega proceso a proceso para las aplicaciones que se ejecutan en

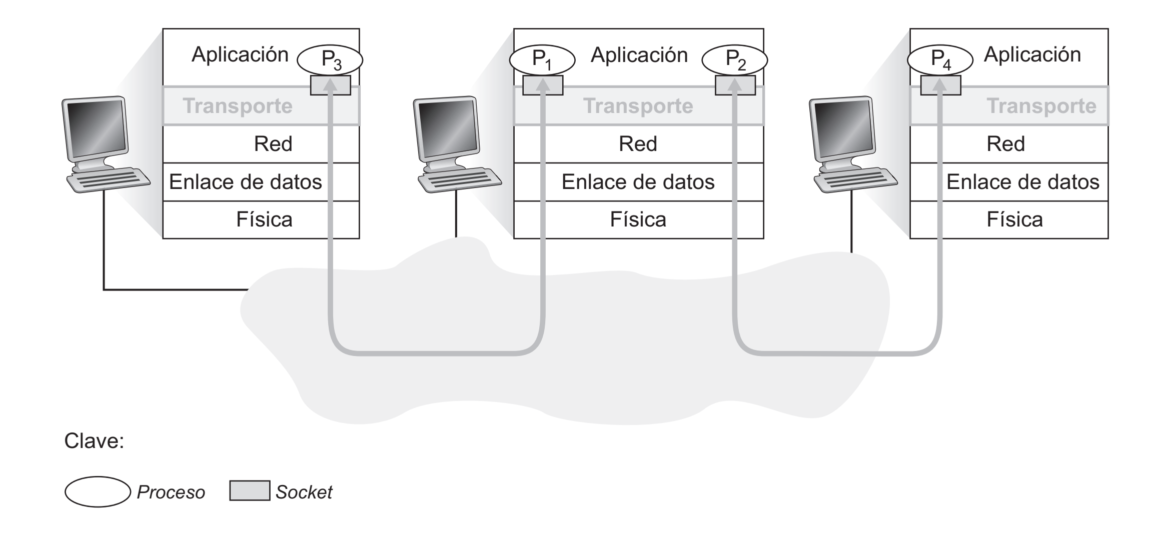


Figura 3.2 • Multiplexación y demultiplexación en la capa de transporte.

los hosts. Con el fin de centrar la explicación, vamos a ver este servicio básico de la capa de transporte en el contexto de Internet. Sin embargo, hay que destacar que un servicio de multiplexación/demultiplexación es necesario en todas las redes de computadoras.

En el host de destino, la capa de transporte recibe segmentos procedentes de la capa de red que tiene justo debajo. La capa de transporte tiene la responsabilidad de entregar los datos contenidos en estos segmentos al proceso de la aplicación apropiada que está ejecutándose en el host. Veamos un ejemplo. Suponga que está sentado frente a su computadora y que está descargando páginas web a la vez que ejecuta una sesión FTP y dos sesiones Telnet. Por tanto, tiene cuatro procesos de aplicación de red en ejecución: dos procesos Telnet, un proceso FTP y un proceso HTTP. Cuando la capa de transporte de su computadora recibe datos procedentes de la capa de red, tiene que dirigir los datos recibidos a uno de estos cuatro procesos. Veamos cómo hace esto.

En primer lugar, recordemos de las Secciones 2.7 y 2.8 que un proceso (como parte de una aplicación de red) puede tener uno o más **sockets**, puertas por las que pasan los datos de la red al proceso, y viceversa. Por tanto, como se muestra en la Figura 3.2, la capa de transporte del host receptor realmente no entrega los datos directamente a un proceso, sino a un socket intermedio. Dado que en cualquier instante puede haber más de un socket en el host receptor, cada socket tiene asociado un identificador único. El formato de este identificador depende de si se trata de un socket UDP o de un socket TCP, como vamos a ver a continuación.

Veamos ahora cómo un host receptor dirige un segmento de entrada de la capa de transporte al socket apropiado. Cada segmento de la capa de transporte contiene un conjunto de campos destinados a este propósito. En el extremo receptor, la capa de transporte examina estos campos para identificar el socket receptor y, a continuación, envía el segmento a dicho socket. Esta tarea de entregar los datos contenidos en un segmento de la capa de transporte al socket correcto es lo que se denomina **demultiplexación**. La tarea de reunir los fragmentos de datos en el host de origen desde los diferentes sockets, encapsulando cada fragmento de datos con la información de cabecera (la cual se utilizará después en el proceso de

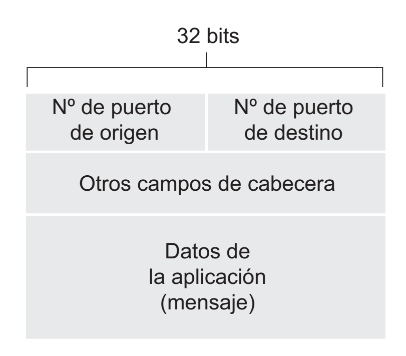


Figura 3.3 • Los campos número de puerto de origen y de destino en un segmento de la capa de transporte.

demultiplexación) para crear los segmentos y pasarlos a la capa de red es lo que se denomina **multiplexación**. Observe que la capa de transporte del host intermedio de la Figura 3.2 tiene que demultiplexar los segmentos que llegan de la capa de red inferior para entregárselos a los procesos P_1 o P_2 de la capa superior; esto se hace dirigiendo los datos del segmento entrante al correspondiente socket del proceso. La capa de transporte del host intermedio también tiene que recopilar los datos salientes desde estos sockets, construir los segmentos de la capa de transporte y pasar dichos segmentos a la capa de red. Aunque hemos presentado la multiplexación y la demultiplexación en el contexto de los protocolos de transporte de Internet, es importante darse cuenta de que esas técnicas son necesarias siempre que un único protocolo en una capa (en la capa de transporte o cualquier otra) sea utilizado por varios protocolos de la capa inmediatamente superior.

Para ilustrar la tarea de demultiplexación, recordemos la analogía doméstica de la seccion anterior. Cada uno de los niños está identificado por su nombre. Cuando Bill recibe un lote de cartas del servicio de correos, lleva a cabo una operación de demultiplexación al determinar a quién van dirigidas las cartas y luego las entrega en mano a sus hermanos. Ann lleva a cabo una operación de multiplexación al recopilar las cartas de sus hermanos y entregar todas las cartas al cartero.

Ahora que ya entendemos las funciones de multiplexación y demultiplexación de la capa de transporte, vamos a examinar cómo se hace esto realmente en un host. Basándonos en las explicaciones anteriores, sabemos que la operación de multiplexación que se lleva a cabo en la capa de transporte requiere (1) que los sockets tengan identificadores únicos y (2) que cada segmento tenga campos especiales que indiquen el socket al que tiene que entregarse el segmento. Estos campos especiales, mostrados en la Figura 3.3, son el **campo número de puerto de origen** y el **campo número de puerto de destino**. (Los segmentos UDP y TCP contienen además otros campos, como veremos en las siguientes secciones del capítulo.) Cada número de puerto es un número de 16 bits comprendido en el rango de 0 a 65535. Los números de puerto pertenecientes al rango de 0 a 1023 se conocen como **números de puertos bien conocidos** y son restringidos, lo que significa que están reservados para ser empleados por los protocolos de aplicación bien conocidos, como por ejemplo HTTP (que utiliza el número de puerto 80) y FTP (que utiliza el número de puerto 21). Puede encontrar la lista de números de puerto bien conocidos en el documento RFC 1700 y su actualización en la dirección http://www.iana.org [RFC 3232]. Al desarrollar una nueva

aplicación (como las aplicaciones desarrolladas en las Secciones 2.7 y 2.8), es necesario asignar un número de puerto a la aplicación.

Ahora ya debería estar claro cómo la capa de transporte *podría* implementar el servicio de demultiplexación: cada socket del host se puede asignar a un número de puerto y, al llegar un segmento al host, la capa de transporte examina el número de puerto de destino contenido en el segmento y lo envía al socket correspondiente. A continuación, los datos del segmento pasan a través del socket y se entregan al proceso asociado. Como veremos, esto es básicamente lo que hace UDP. Sin embargo, también comprobaremos que la tarea de multiplexación/demultiplexación en TCP es más sutil.

Multiplexación y demultiplexación sin conexión

Recordemos de la Sección 2.8 que un programa Java que se ejecuta en un host puede crear un socket UDP mediante la línea de código:

```
DatagramSocket miSocket = new DatagramSocket();
```

Cuando se crea un socket UDP de este modo, la capa de transporte asigna automáticamente un número de puerto al socket. En particular, la capa de transporte asigna un número de puerto comprendido en el rango de 1024 a 65535 que actualmente no esté siendo utilizado en ese host por ningún otro puerto UDP. Alternativamente, un programa Java podría crear un socket ejecutando la línea de código:

```
DatagramSocket miSocket = new DatagramSocket(19157);
```

En este caso, la aplicación asigna un número de puerto específico, por ejemplo, el 19157, al socket UDP. Si el desarrollador de la aplicación que escribe el código estuviera implementando el lado del servidor de un "protocolo bien conocido", entonces tendría que asignar el correspondiente número de puerto bien conocido. Normalmente, el lado del cliente de la aplicación permite a la capa de transporte asignar de forma automática (y transparente) el número de puerto, mientras que el lado de servidor de la aplicación asigna un número de puerto específico.

Una vez asignados los números de puerto a los sockets UDP, podemos describir de forma precisa las tareas de multiplexación y demultiplexación en UDP. Suponga que un proceso del host A, con el puerto UDP 19157, desea enviar un fragmento de datos de una aplicación a un proceso con el puerto UDP 46428 en el host B. La capa de transporte del host A crea un segmento de la capa de transporte que incluye los datos de aplicación, el número de puerto de origen (19157), el número de puerto de destino (46428) y otros dos valores (que veremos más adelante, pero que por el momento no son importantes para el tema que nos ocupa). La capa de transporte pasa a continuación el segmento resultante a la capa de red. La capa de red encapsula el segmento en un datagrama IP y hace el máximo esfuerzo por entregar el segmento al host receptor. Si el segmento llega al host receptor B, la capa de transporte del mismo examina el número de puerto de destino especificado en el segmento (46428) y entrega el segmento a su socket identificado por el puerto 46428. Observe que el host B podría estar ejecutando varios procesos, cada uno de ellos con su propio socket UDP y número de puerto asociado. A medida que los segmentos UDP llegan de la red, el host B dirige (demultiplexa) cada segmento al socket apropiado examinando el número de puerto de destino del segmento.

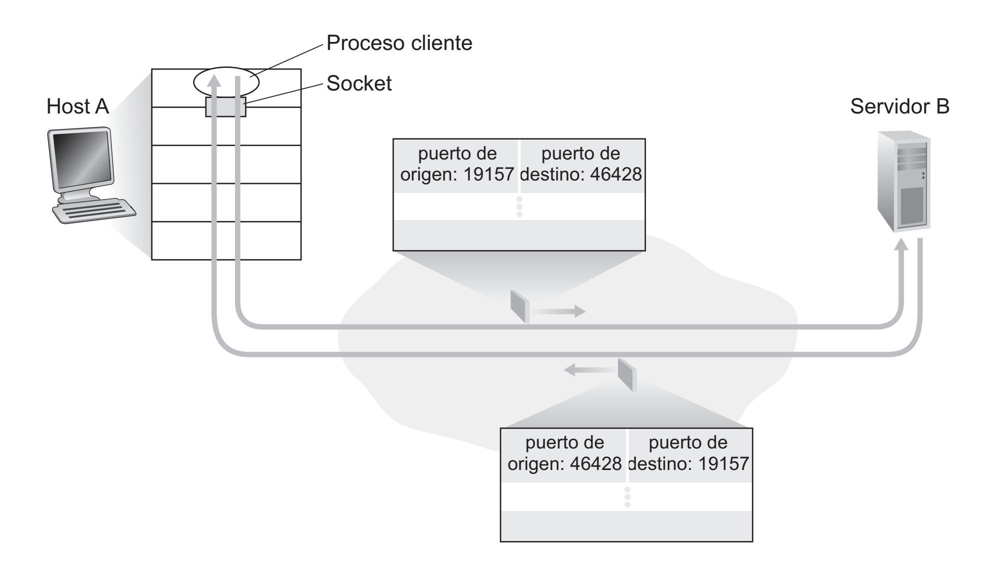


Figura 3.4 • Inversión de los números de puerto de origen y de destino.

Es importante observar que un socket UDP queda completamente identificado por una tupla que consta de una dirección IP de destino y un número de puerto de destino. En consecuencia, si dos segmentos UDP tienen diferentes direcciones IP y/o números de puerto de origen, pero la misma dirección IP de *destino* y el mismo número puerto de *destino,* entonces los dos segmentos se enviarán al mismo proceso de destino a través del mismo socket de destino.

Es posible que se esté preguntando en este momento cuál es el propósito del número de puerto de origen. Como se muestra en la Figura 3.4, en el segmento A a B, el número de puerto de origen forma parte de una "dirección de retorno"; es decir, si B desea devolver un segmento a A, el puerto de destino en el segmento de B a A tomará su valor del valor del puerto de origen del segmento de A a B. (La dirección de retorno completa es el número de puerto de origen y la dirección IP de A.) Por ejemplo, recuerde el programa de servidor UDP estudiado en la Sección 2.8. En `UDPServer.java`, el servidor utiliza un método para extraer el número de puerto de origen del segmento que recibe del cliente; a continuación, envía un segmento nuevo al cliente, utilizando como número de puerto de destino el número de puerto de origen extraído del segmento recibido.

Multiplexación y demultiplexación orientadas a la conexión

Para entender la demultiplexación TCP, tenemos que tener en cuenta los sockets TCP y el establecimiento de las conexiones TCP. Una sutil diferencia entre un socket TCP y un socket UDP es que el primero queda identificado por una tupla de cuatro elementos: dirección IP de origen, número de puerto de origen, dirección IP de destino, número de puerto de destino. Por tanto, cuando un segmento TCP llega a un host procedente de la red, el host emplea los cuatro valores para dirigir (demultiplexar) el segmento al socket apropiado. En particular, y al contrario de lo que ocurre con UDP, dos segmentos TCP entrantes con direcciones IP de origen o números de puerto de origen diferentes (con la excepción de un segmento TCP que transporte la solicitud original de establecimiento de conexión) serán dirigidos a

dos sockets distintos. Con el fin de profundizar un poco, consideremos de nuevo el ejemplo de programación cliente-servidor TCP de la Sección 2.7:

- La aplicación de servidor TCP tiene un “socket de acogida”, que está a la espera de las solicitudes de establecimiento de conexión procedentes de los clientes TCP en el puerto 6789 (véase la Figura 2.29).
- El cliente TCP genera un segmento de establecimiento de conexión con la línea de código:

```
Socket socketCliente = new Socket("nombreHostServidor", 6789);
```

- Una solicitud de establecimiento de conexión no es nada más que un segmento TCP con el número de puerto de destino 6789 y un conjunto especial de bits de establecimiento de conexión en la cabecera TCP (que veremos en la Sección 3.5). El segmento también incluye un número de puerto de origen, que habrá sido seleccionado por el cliente. La línea de código anterior también crea un socket TCP para el proceso cliente a través del cual los datos pueden entrar y salir del proceso cliente.
- Cuando el sistema operativo del host que está ejecutando el proceso servidor recibe el segmento de entrada de solicitud de conexión con el puerto de destino 6789, localiza el proceso de servidor que está esperando para aceptar una conexión en el número de puerto 6789. El proceso de servidor crea entonces un nuevo socket:

```
Socket socketConexion = socketAcogida.accept();
```

- Además, la capa de transporte en el servidor toma nota de los cuatro valores siguientes contenidos en el segmento de solicitud de conexión: (1) el número de puerto de origen en el segmento, (2) la dirección IP del host de origen, (3) el número de puerto de destino en el segmento y (4) su propia dirección IP. El socket de conexión recién creado queda identificado por estos cuatro valores; así, todos los segmentos que lleguen después y cuyo puerto de origen, dirección IP de origen, puerto de destino y dirección IP de destino se correspondan con estos cuatro valores serán enviados a este socket. Una vez establecida la conexión TCP, el cliente y el servidor podrán enviarse datos entre sí.

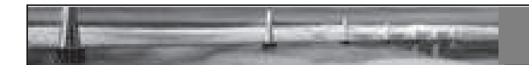
El host servidor puede dar soporte a muchos sockets TCP simultáneos, estando cada socket asociado a un proceso y con cada socket identificado por su tupla de cuatro elementos. Cuando un segmento TCP llega al host, los cuatro campos (dirección IP de origen, puerto de origen, dirección IP de destino y puerto de destino) se utilizan para dirigir (demultiplexar) el segmento al socket apropiado.

Esta situación se ilustra en la Figura 3.5, en la que el host C inicia dos sesiones HTTP con el servidor B y el host A inicia una sesión HTTP también con B. Los hosts A y C y el servidor B tienen sus propias direcciones IP únicas (A, C y B, respectivamente). El host C asigna dos números de puerto de origen diferentes (26145 y 7532) a sus dos conexiones HTTP. Dado que el host A está seleccionando los números de puerto de origen independientemente de C, también puede asignar el número de puerto de origen 26145 a su conexión HTTP. Pero esto no es un problema: el servidor B todavía podrá demultiplexar correctamente las dos conexiones con el mismo número de puerto de origen, ya que tienen direcciones IP de origen diferentes.

Servidores web y TCP

Antes de dar por terminada esta sección, es interesante comentar algunas cosas acerca de los servidores web y de cómo utilizan los números de puerto. Considere un host que está ejecutando un servidor web, como por ejemplo un servidor web Apache en el puerto 80. Cuando los clientes (por ejemplo, los navegadores) envían segmentos al servidor, *todos* los segmentos tendrán el puerto de destino 80. En particular, tanto los segmentos para el establecimiento de la conexión inicial como los segmentos que transportan los mensajes de solicitud HTTP utilizarán como puerto de destino el puerto 80. Como acabamos de describir, el servidor diferencia los segmentos procedentes de los distintos clientes mediante las direcciones IP de origen y los números de puerto de origen.

La Figura 3.5 nos muestra un servidor web que genera un nuevo proceso para cada conexión. Como se muestra en esta figura, cada uno de estos procesos tiene su propio socket de conexión a través del cual llegan las solicitudes HTTP y se envían las respuestas HTTP. Sin embargo, también hemos mencionado que no existe siempre una correspondencia uno a



SEGURIDAD

EXPLORACIÓN DE PUERTOS

Hemos visto que un proceso servidor espera pacientemente en un puerto abierto a ser contactado por un cliente remoto. Algunos puertos están reservados para aplicaciones bien conocidas (como por ejemplo servidores web, FTP, DNS y SMTP); otros puertos, por convenio, son utilizados por aplicaciones populares (como por ejemplo Microsoft 2000 SQL Server que está a la escucha en el puerto UDP 1434). Por tanto, si determinamos que un puerto está abierto en un host, podemos ser capaces de hacer corresponder dicho puerto a una aplicación específica que se ejecute en el host. Esto es muy útil para los administradores de sistemas, ya que suelen estar interesados en saber qué aplicaciones de red están ejecutándose en los hosts de sus redes. Pero los atacantes, con el fin de detectar "las brechas en el muro", también desean saber qué puertos están abiertos en los hosts objetivo. Si se localiza un host que está ejecutando una aplicación con un defecto de seguridad conocido (por ejemplo, si un servidor SQL que está a la escucha en el puerto 1434 fuera objeto de un desbordamiento de buffer, permitiendo a un usuario remoto ejecutar código arbitrario en el host vulnerable, un defecto explotado por el gusano Slammer [CERT 2003-04]), entonces dicho host estaría en condiciones para recibir un ataque.

Determinar qué aplicaciones están a la escucha en qué puertos es una tarea relativamente fácil. Además, existen numerosos programas de dominio público, conocidos como escáneres de puertos, que realizan este trabajo. Quizá el más ampliamente utilizado de estos programas es nmap, que está disponible gratuitamente en http://insecure.org/nmap y se incluye en la mayor parte de las distribuciones de Linux. Para TCP, nmap explora secuencialmente los puertos buscando puertos que acepten conexiones TCP. Para UDP, nmap también explora secuencialmente los puertos buscando puertos UDP que respondan a segmentos UDP transmitidos. En ambos casos, nmap devuelve una lista de puertos abiertos, cerrados o inalcanzables. Un host que ejecute nmap puede tratar de explorar cualquier host objetivo situado *en cualquier lugar* de Internet. En la Sección 3.5.6 volveremos a hablar de nmap, al tratar la gestión de las conexiones TCP.

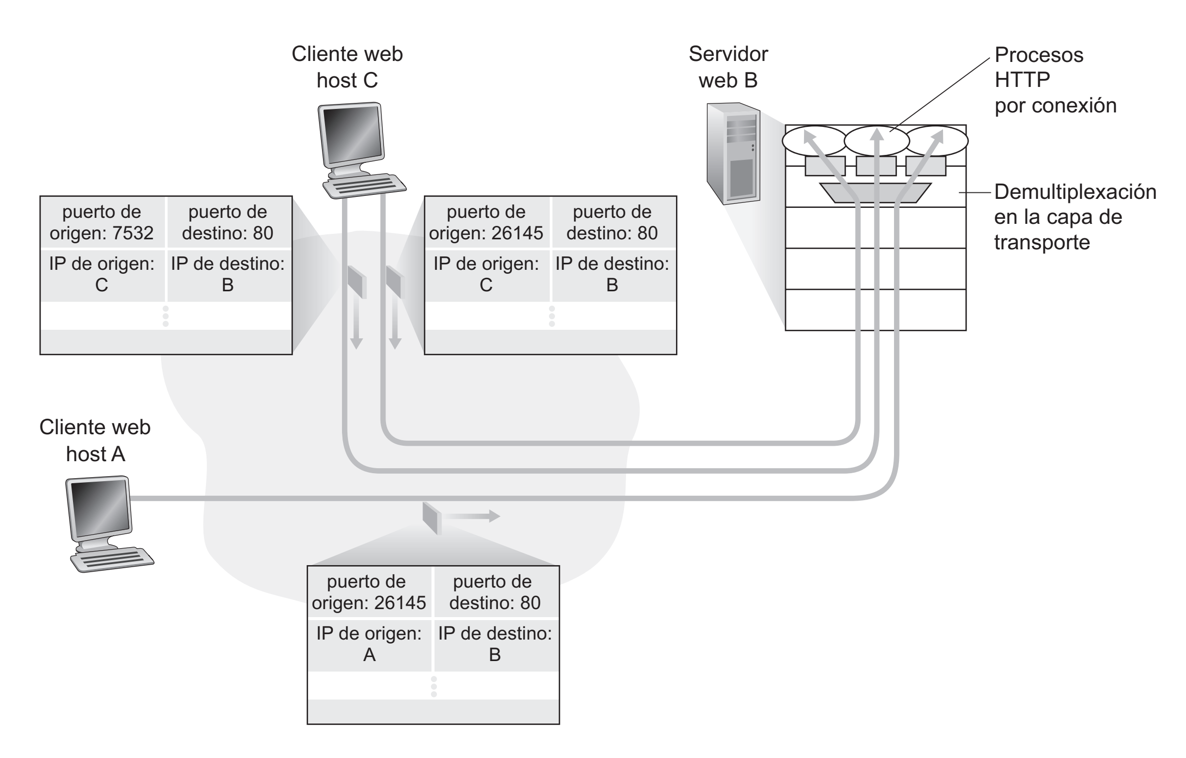


Figura 3.5 • Dos clientes utilizando el mismo número de puerto de destino (80) para comunicarse con la misma aplicación de servidor web.

uno entre los sockets de conexión y los procesos. De hecho, los servidores web actuales de altas prestaciones a menudo sólo utilizan un proceso y crean una nueva hebra con un nuevo socket de conexión para cada nueva conexión de un cliente (una hebra es una especie de subproceso de baja complejidad). Si realizó la primera tarea de programación del Capítulo 2, habrá creado un servidor web que hace exactamente esto. En un servidor así, en cualquier instante puede haber muchos sockets de conexión (con distintos identificadores) asociados al mismo proceso.

Si el cliente y el servidor están utilizando HTTP persistente, entonces mientras dure la conexión persistente el cliente y el servidor intercambiarán mensajes HTTP a través del mismo socket de servidor. Sin embargo, si el cliente y el servidor emplean HTTP no persistente, entonces se creará y cerrará una nueva conexión TCP para cada pareja solicitud/respuesta y, por tanto, se creará (y luego se cerrará) un nuevo socket para cada solicitud/respuesta. Esta creación y cierre de sockets tan frecuente puede afectar seriamente al rendimiento de un servidor web ocupado (aunque se pueden utilizar una serie de trucos del sistema operativo para mitigar el problema). Los lectores interesados en los problemas que el uso de HTTP persistente y no persistente plantea a los sistemas operativos pueden consultar [Nielsen 1997; Nahum 2002].

Ahora que ya hemos visto en qué consiste la multiplexación y la demultiplexación en la capa de transporte, podemos pasar a tratar uno de los protocolos de transporte de Internet: UDP. En la siguiente sección veremos que UDP añade algo más al protocolo de la capa de red que un servicio de multiplexación/demultiplexación.

vacuo: superficial, que carece de contenido o interés

3.3 Transporte sin conexión: UDP

En esta sección vamos a ocuparnos de UDP, vamos a ver cómo funciona y qué hace. Le animamos a releer la Sección 2.1, en la que se incluye una introducción al modelo de servicio de UDP y la Sección 2.8, en la que se explica la programación de sockets con UDP.

Para iniciar nuestro análisis de UDP, suponga que queremos diseñar un protocolo de transporte simple y poco sofisticado. ¿Cómo haríamos esto? En primer lugar, podemos considerar la utilización de un protocolo de transporte vacuo. Es decir, en el lado emisor, tomará los mensajes del proceso de la aplicación y los pasará directamente a la capa de red; en el lado de recepción, tomará los mensajes procedentes de la capa de red y los pasará directamente al proceso de la aplicación. Pero, como hemos aprendido en la sección anterior, hay algunas cosas mínimas que tenemos que hacer. Como mínimo, la capa de transporte tiene que proporcionar un servicio de multiplexación/demultiplexación que permita transferir los datos entre la capa de red y el proceso de la capa de aplicación correcto.

UDP, definido en el documento [RFC 768], hace casi lo mínimo que un protocolo de transporte debe hacer. Además de la función de multiplexación/demultiplexación y de algún mecanismo de comprobación de errores, no añade nada a IP. De hecho, si el desarrollador de la aplicación elige UDP en lugar de TCP, entonces prácticamente es la aplicación la que se comunica directamente con IP. UDP toma los mensajes procedentes del proceso de la aplicación, asocia los campos correspondientes a los números de puerto de origen y de destino para proporcionar el servicio de multiplexación/demultiplexación, añade dos campos pequeños más y pasa el segmento resultante a la capa de red. La capa de red encapsula el segmento de la capa de transporte en un datagrama IP y luego hace el mejor esfuerzo por entregar el segmento al host receptor. Si el segmento llega al host receptor, UDP utiliza el número de puerto de destino para entregar los datos del segmento al proceso apropiado de la capa de aplicación. Observe que con UDP no tiene lugar una fase de establecimiento de la conexión entre las entidades de la capa de transporte emisora y receptora previa al envío del segmento. Por esto, se dice que UDP es un protocolo *sin conexión.*

DNS es un ejemplo de un protocolo de la capa de aplicación que habitualmente utiliza UDP. Cuando la aplicación DNS de un host desea realizar una consulta, construye un mensaje de consulta DNS y lo pasa a UDP. Sin llevar a cabo ningún proceso para establecer una conexión con la entidad UDP que se ejecuta en el sistema terminal de destino, el protocolo UDP del lado del host añade campos de cabecera al mensaje y pasa el segmento resultante a la capa de red, la cual encapsula el segmento UDP en un datagrama y lo envía a un servidor de nombres. La aplicación DNS que se ejecuta en el host que ha hecho la consulta espera entonces hasta recibir una respuesta a su consulta. Si no la recibe (posiblemente porque la red subyacente ha perdido la consulta o la respuesta), bien intenta enviar la consulta a otro servidor de nombres o bien informa a la aplicación invocante de que no puede obtener una respuesta.

Es posible que ahora se esté preguntando por qué un desarrollador de aplicaciones podría decidir crear una aplicación sobre UDP en lugar de sobre TCP. ¿No sería preferible emplear siempre TCP, puesto que proporciona un servicio de transferencia de datos fiable y UDP no? La respuesta es no, ya que muchas aplicaciones están mejor adaptadas a UDP por las siguientes razones:

- *Mejor control en el nivel de aplicación sobre qué datos se envían y cuándo*. Con UDP, tan pronto como un proceso de la capa de aplicación pasa datos a UDP, UDP los empa-

queta en un segmento UDP e inmediatamente entrega el segmento a la capa de red. Por el contrario, TCP dispone de un mecanismo de control de congestión que regula el flujo del emisor TCP de la capa de transporte cuando uno o más de los enlaces existentes entre los hosts de origen y de destino están excesivamente congestionados. TCP también continuará reenviando un segmento hasta que la recepción del mismo haya sido confirmada por el destino, independientemente de cuánto se tarde en llevar a cabo esta entrega fiable. Puesto que las aplicaciones en tiempo real suelen requerir una velocidad mínima de transmisión, no permiten un retardo excesivo en la transmisión de los segmentos y pueden tolerar algunas pérdidas de datos, el modelo de servicio de TCP no se adapta demasiado bien a las necesidades de este tipo de aplicaciones. Como veremos más adelante, estas aplicaciones pueden utilizar UDP e implementar, como parte de la aplicación, cualquier funcionalidad adicional que sea necesaria más allá del servicio básico de entrega de segmentos de UDP.

- *Sin establecimiento de la conexión.* Como se explicará más adelante, TCP lleva a cabo un proceso de establecimiento de la conexión en tres fases antes de iniciar la transferencia de datos. UDP inicia la transmisión sin formalidades preliminares. Por tanto, UDP no añade ningún retardo a causa del establecimiento de una conexión. Probablemente, ésta es la razón principal por la que DNS opera sobre UDP y no sobre TCP (DNS sería mucho más lento si se ejecutara sobre TCP). HTTP utiliza TCP en lugar de UDP, ya que la fiabilidad es crítica para las páginas web con texto. Pero, como hemos comentado brevemente en la Sección 2.2, el retardo debido al establecimiento de la conexión TCP en HTTP contribuye de forma notable a los retardos asociados con la descarga de documentos web.
- *Sin información del estado de la conexión.* TCP mantiene información acerca del estado de la conexión en los sistemas terminales. En el estado de la conexión se incluye información acerca de los buffers de recepción y envío, de los parámetros de control de congestión y de los parámetros relativos al número de secuencia y de reconocimiento. En la Sección 3.5 veremos que esta información de estado es necesaria para implementar el servicio fiable de transferencia de datos y proporcionar los mecanismos de control de congestión de TCP. Por el contrario, UDP no mantiene información del estado de la conexión y no controla ninguno de estos parámetros. Por esta razón, un servidor dedicado a una aplicación concreta suele poder soportar más clientes activos cuando la aplicación se ejecuta sobre UDP que cuando lo hace sobre TCP.
- *Poca sobrecarga debida a la cabecera de los paquetes.* Los segmentos TCP contienen 20 bytes en la cabecera de cada segmento, mientras que UDP sólo requiere 8 bytes.

La tabla de la Figura 3.6 enumera aplicaciones de Internet muy populares junto con los protocolos de transporte que utilizan. Como era de esperar, el correo electrónico, el acceso remoto a terminales, la Web y la transferencia de archivos se ejecutan sobre TCP, ya que todas estas aplicaciones necesitan el servicio fiable de transferencia de datos de TCP. No obstante, muchas aplicaciones importantes se ejecutan sobre UDP en lugar de sobre TCP. UDP se utiliza para las actualizaciones de las tablas de enrutamiento RIP (véase la Sección 4.6.1). Puesto que las actualizaciones RIP se envían periódicamente (normalmente cada cinco minutos), las actualizaciones perdidas se reemplazan por otras más recientes, haciendo inútiles las actualizaciones perdidas ya desactualizadas. UDP también se emplea para transmitir los datos de administración de la red (SNMP; véase el Capítulo 9). En este caso,

Aplicación	Protocolo de la capa de aplicación	Protocolo de transporte subyacente
Correo electrónico	SMTP	TCP
Acceso a terminales remotos	Telnet	TCP
Web	HTTP	TCP
Transferencia de archivos	FTP	TCP
Servidor de archivos remoto	NFS	Normalmente UDP
Flujos multimedia	Normalmente propietario	UDP o TCP
Telefonía por Internet	Normalmente propietario	UDP o TCP
Administración de red	SNMP	Normalmente UDP
Protocolo de enrutamiento	RIP	Normalmente UDP
Traducción de nombres	DNS	Normalmente UDP

Figura 3.6 • Aplicaciones de Internet populares y sus protocolos de transporte subyacentes.

UDP es preferible a TCP, ya que las aplicaciones de administración de la red a menudo se tienen que ejecutar cuando la red se encuentra en un estado de sobrecarga (precisamente en las situaciones en las que es difícil realizar transferencias de datos fiables y con control de la congestión). Además, como ya hemos mencionado anteriormente, DNS se ejecuta sobre UDP, evitando de este modo los retardos de establecimiento de la conexión TCP.

Como se indica en la Figura 3.6, actualmente tanto UDP como TCP se utilizan con aplicaciones multimedia, como la telefonía por Internet, las videoconferencias en tiempo real y la reproducción de flujos de vídeos y audios almacenados. En el Capítulo 7 nos ocuparemos de estas aplicaciones. Por el momento, basta con mencionar que todas estas aplicaciones toleran la pérdida de una pequeña cantidad de paquetes, por lo que una transferencia de datos fiable no es absolutamente crítica para que la aplicación funcione correctamente. Además, las aplicaciones en tiempo real, como la telefonía por Internet y la videoconferencia, responden muy mal a los mecanismos de control de congestión de TCP. Por estas razones, los desarrolladores de aplicaciones multimedia pueden elegir ejecutar sus aplicaciones sobre UDP y no sobre TCP. No obstante, cada vez se utiliza más TCP para el transporte de flujos multimedia. Por ejemplo, puede leer en [Sripanidkulchai 2004] que casi el 75% de los flujos multimedia en directo y a la carta utilizan TCP. Cuando la tasa de pérdidas de paquetes es baja y teniendo en cuenta que algunas organizaciones bloquean el tráfico UDP por razones de seguridad (véase el Capítulo 8), TCP se está convirtiendo en un protocolo cada vez más atractivo para el transporte de flujos multimedia.

Aunque hoy en día es común emplear UDP para ejecutar las aplicaciones multimedia, continúa siendo un tema controvertido. Como ya hemos dicho, UDP no proporciona mecanismos de control de congestión, y estos mecanismos son necesarios para impedir que la red entre en un estado de congestión en el que se realice muy poco trabajo útil. Si todo el mundo deseara reproducir flujos de vídeo a alta velocidad sin utilizar ningún mecanismo de control

de congestión, se produciría tal desbordamiento de paquetes en los routers que muy pocos paquetes UDP lograrían recorrer con éxito la ruta entre el origen y el destino. Además, las altas tasas de pérdidas inducidas por los emisores UDP no controlados harían que los emisores TCP (que, como veremos, reducen sus velocidades de transmisión para hacer frente a la congestión) disminuyeran dramáticamente sus velocidades. Por tanto, la ausencia de un mecanismo de control de congestión en UDP puede dar lugar a altas tasas de pérdidas entre un emisor y un receptor UDP y al estrangulamiento de las sesiones TCP, lo cual es un problema potencialmente serio [Floyd 1999]. Muchos investigadores han propuesto nuevos mecanismos para forzar a todas las fuentes de datos, incluyendo las de tipo UDP, a llevar a cabo un control adaptativo de la congestión [Mahdavi 1997; Floyd 2000; Kohler 2006; RFC 4340].

Antes de pasar a examinar la estructura de los segmentos UDP, tenemos que comentar que *es* posible que una aplicación disponga de un servicio fiable de transferencia de datos utilizando UDP. Esto puede conseguirse si las características de fiabilidad se incorporan a la propia aplicación (por ejemplo, añadiendo mecanismos de reconocimiento y retransmisión, tales como los que vamos a ver en la siguiente sección). Pero se trata de una tarea nada sencilla que requiere una intensa tarea de depuración de las aplicaciones. No obstante, incorporar los mecanismos de fiabilidad directamente en la aplicación permite que ella misma "selo guise y se lo coma". Es decir, los procesos de aplicación pueden comunicarse de forma fiable sin estar sujetos a las restricciones de la velocidad de transmisión impuestas por el mecanismo de control de congestión de TCP.

3.3.1 Estructura de los segmentos UDP

La estructura de los segmentos UDP, mostrada en la Figura 3.7, está definida en el documento RFC 768. Los datos de la aplicación ocupan el campo de datos del segmento UDP. Por ejemplo, para DNS el campo de datos contiene un mensaje de consulta o un mensaje de respuesta. En el caso de una aplicación de flujo de audio, las muestras del audio llenarán el campo de datos. La cabecera UDP sólo tiene cuatro campos, y cada uno de ellos tiene una longitud de dos bytes. Como se ha explicado en la sección anterior, los números de puerto permiten al host de destino pasar los datos de la aplicación al proceso apropiado que está ejecutándose en el sistema terminal de destino (es decir, realizar la función de demultiplexación). El host receptor utiliza la suma de comprobación para detectar si se han introducido errores en el segmento. En realidad, la suma de comprobación también se calcula para unos pocos campos de la cabecera IP, además de para el segmento UDP. Pero por el momento no vamos a tener en cuenta este detalle para ver el bosque a través de los árboles. Veamos ahora cómo se calcula la suma de comprobación. En la Sección 5.2 se describen los principios básicos para la detección de errores. El campo longitud especifica la longitud del segmento UDP en bytes, incluyendo la cabecera.

3.3.2 Suma de comprobación de UDP

La suma de comprobación de UDP proporciona un mecanismo de detección de errores. Es decir, se utiliza para determinar si los bits contenidos en el segmento UDP han sido alterados según se desplazaban desde el origen hasta el destino (por ejemplo, a causa de la existencia de ruido en los enlaces o mientras estaban almacenados en un router). UDP en el lado del emisor calcula el complemento a 1 de la suma de todas las palabras de 16 bits del seg-

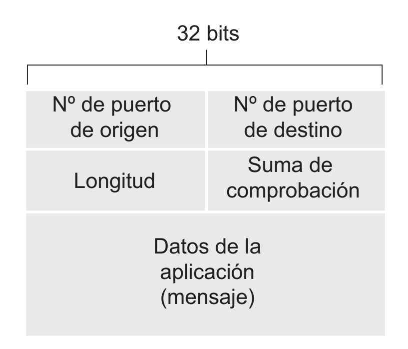


Figura 3.7 • Estructura de un segmento UDP.

mento, acarreando cualquier desbordamiento obtenido durante la operación de suma sobre el bit de menor peso. Este resultado se almacena en el campo suma de comprobación del segmento UDP. He aquí un ejemplo sencillo de cálculo de una suma de comprobación. Puede obtener información detallada acerca de una implementación eficiente del cálculo en el RFC 1071, así como de su rendimiento con datos reales en [Stone 1998; Stone 2000]. Por ejemplo, suponga que tenemos las siguientes tres palabras de 16 bits:

0110011001100000
0101010101010101
1000111100001100

La suma de las dos primeras palabras de 16 bits es:

0110011001100000
<u>0101010101010101</u>
1011101110110101

Sumando la tercera palabra a la suma anterior, obtenemos,

1011101110110101
<u>1000111100001100</u>
0100101011000010

Observe que en esta última suma se produce un desbordamiento, el cual se acarrea sobre el bit de menor peso. El complemento a 1 se obtiene convirtiendo todos los 0 en 1 y todos los 1 en 0. Por tanto, el complemento a 1 de la suma 0100101011000010 es 1011010100111101, que es la suma de comprobación. En el receptor, las cuatro palabras de 16 bits se suman, incluyendo la suma de comprobación. Si no se han introducido errores en el paquete, entonces la suma en el receptor tiene que ser 1111111111111111. Si uno de los bits es un 0, entonces sabemos que el paquete contiene errores.

Es posible que se esté preguntando por qué UDP proporciona una suma de comprobación, cuando muchos protocolos de la capa de enlace (incluyendo el popular protocolo Ethernet) también proporcionan mecanismos de comprobación de errores. La razón de ello es que no existe ninguna garantía de que todos los enlaces existentes entre el origen y el des-

tino proporcionen un mecanismo de comprobación de errores; es decir, uno de los enlaces puede utilizar un protocolo de la capa de enlace que no proporcione comprobación de errores. Además, incluso si los segmentos se transfieren correctamente a través del enlace, es posible que se introduzcan errores de bit cuando un segmento se almacena en la memoria de un router. Dado que no están garantizadas ni la fiabilidad enlace a enlace, ni la detección de errores durante el almacenamiento en memoria, UDP tiene que proporcionar un mecanismo de detección de errores en la capa de transporte, *terminal a terminal*, si el servicio de transferencia de datos terminal a terminal ha de proporcionar la de detección de errores. Este es un ejemplo del famoso **principio terminal a terminal** del diseño de sistemas [Saltzer 1984], que establece que como cierta funcionalidad (en este caso, la detección de errores) debe implementarse terminal a terminal, "las funciones incluidas en los niveles inferiores pueden ser redundantes o escasamente útiles si se comparan con el coste de proporcionarlas en el nivel superior".

Dado que IP está pensado para ejecutarse sobre prácticamente cualquier protocolo de la capa 2, resulta útil para la capa de transporte proporcionar un mecanismo de comprobación de errores como medida de seguridad. Aunque UDP proporciona un mecanismo de comprobación de errores, no hace nada para recuperarse del error. Algunas implementaciones de UDP simplemente descartan el segmento dañado y otras lo pasan a la aplicación junto con una advertencia.

Hasta aquí llega nuestra exposición sobre UDP. Pronto veremos que TCP ofrece a las aplicaciones un servicio de transferencia de datos fiable, así como otros servicios que UDP no proporciona. Naturalmente, TCP también es más complejo que UDP. Sin embargo, antes de abordar TCP, nos resultará útil volver unos pasos atrás y ocuparnos primero de los principios que subyacen a una transferencia de datos fiable.



TP9

3.4 Principios de un servicio de transferencia de datos fiable

En esta sección vamos a considerar el problema de la transferencia de datos fiable en un contexto general. Este enfoque es conveniente porque el problema de implementar servicios de transferencia de datos fiables no sólo aparece en la capa de transporte, sino también en la capa de enlace y en la capa de aplicación. El problema general tiene por tanto una gran relevancia en las redes de computadoras. En efecto, si tuviéramos que identificar la lista de los diez problemas más importantes que afectan a las redes, éste sería un candidato a liderar dicha lista. En la siguiente sección examinaremos TCP y, en concreto, mostraremos que TCP aplica muchos de los principios que vamos a describir.

La Figura 3.8 ilustra el marco de trabajo que vamos a emplear en nuestro estudio sobre la transferencia de datos fiable. La abstracción del servicio proporcionada a las entidades de la capa superior es la de un canal fiable a través del cual se pueden transferir datos. Disponiendo de un canal fiable, ninguno de los bits de datos transferidos está corrompido (cambia de 0 a 1, o viceversa) ni se pierde, y todos se entregan en el orden en que fueron enviados. Éste es precisamente el modelo de servicio ofrecido por TCP a las aplicaciones de Internet que lo invocan.

Es la responsabilidad de un **protocolo de transferencia de datos fiable** implementar esta abstracción del servicio. Esta tarea es complicada por el hecho de que la capa que hay por

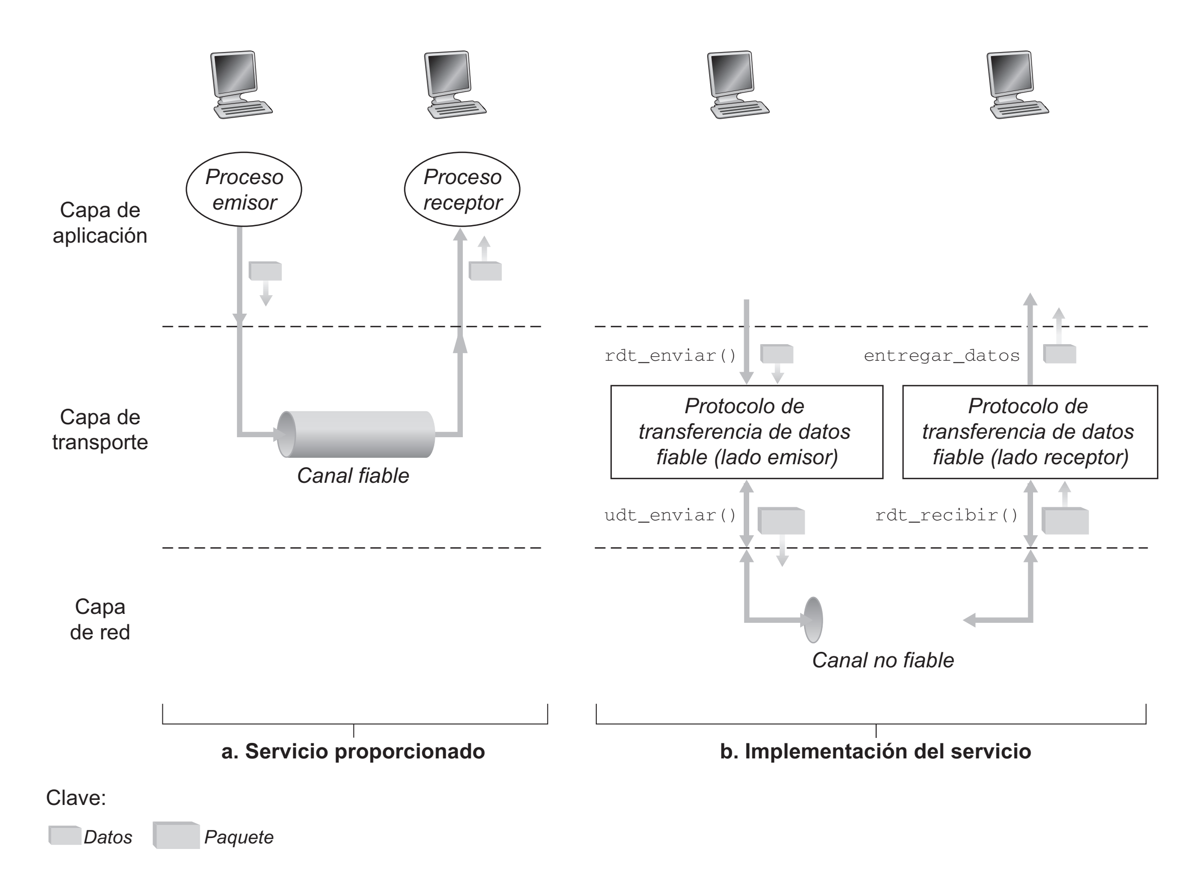


Figura 3.8 • Transferencia de datos fiable: modelo del servicio e implementación del servicio.

debajo del protocolo de transferencia de datos puede ser no fiable. Por ejemplo, TCP es un protocolo de transferencia de datos fiable que se implementa encima de una capa de red terminal a terminal no fiable (IP). De forma más general, la capa que hay debajo de los dos puntos terminales que se comunican de forma fiable puede ser un único enlace físico (como en el caso de un protocolo de transferencia de datos a nivel de enlace) o una interred global (como en el caso de un protocolo del nivel de transporte). Para nuestros propósitos, sin embargo, podemos considerar esta capa inferior simplemente como un canal punto a punto no fiable.

En esta sección vamos a desarrollar de forma incremental los lados del emisor y del receptor de un protocolo de transferencia de datos fiable, considerando modelos cada vez más complejos del canal subyacente. La Figura 3.8(b) ilustra las interfaces de nuestro protocolo de transferencia de datos. El lado emisor del protocolo de transferencia de datos será invocado desde la capa superior mediante una llamada a `rdt_enviar()`, que pasará los datos que haya que entregar a la capa superior en el lado receptor. Aquí `rdt` hace referencia al protocolo de transferencia de datos fiable (*reliable data transfer*) y `_enviar` indica que el lado emisor de `rdt` está siendo llamado. (¡El primer paso para desarrollar un buen protocolo es elegir un buen nombre!) En el lado receptor, `rdt_recibir()` será llamado cuando llegue un paquete desde el lado receptor del canal. Cuando el protocolo `rdt` desea suministrar datos a la capa superior, lo hará llamando a `entregar_datos()`. De aquí en adelante

utilizaremos el término “paquete” en lugar de “segmento” de la capa de transporte. Puesto que la teoría desarrollada en esta sección se aplica a las redes de computadoras en general y no sólo a la capa de transporte de Internet, quizá resulte más apropiado el término genérico “paquete”.

En esta sección únicamente consideraremos el caso de la **transferencia de datos unidireccional**, es decir, los datos se transfieren desde el lado emisor al receptor. El caso de la **transferencia de datos bidireccional** (es decir, full-duplex) conceptualmente no es más difícil, pero es considerablemente más tediosa de explicar. Aunque sólo abordemos la transferencia de datos unidireccional, tendremos en cuenta que los lados emisor y receptor de nuestro protocolo necesitan transmitir paquetes en *ambas* direcciones, como se indica en la Figura 3.8. Veremos brevemente que, además de intercambiar paquetes que contengan los datos que van a transferirse, los lados emisor y receptor de `rdt` también intercambian paquetes de control de una parte a otra. Ambos lados, emisor y receptor, de `rdt` envían paquetes al otro lado haciendo una llamada a `udt_enviar()` (donde `udt` hace referencia a una transferencia de datos no fiable [*unreliable data transfer*]).

3.4.1 Construcción de un protocolo de transferencia de datos fiable

Ahora vamos a ver una serie de protocolos de complejidad creciente, hasta llegar a un protocolo de transferencia de datos fiable sin defectos.

Transferencia de datos fiable sobre un canal totalmente fiable: rdt1.0

En primer lugar consideremos el caso más simple, en el que el canal subyacente es completamente fiable. El protocolo en sí, que denominaremos `rdt1.0`, es trivial. En la Figura 3.9 se muestran las definiciones de las **máquinas de estados finitos (FSM, *Finite-State Machine*)** para el emisor y el receptor de `rdt1.0`. La máquina de estados finitos de la Figura 3.9(a) define el funcionamiento del emisor, mientras que la FSM de la Figura 3.9(b) define el funcionamiento del receptor. Es importante observar que existen máquinas de estados finitos separadas para el emisor y el receptor. Cada una de las máquinas de esta figura tiene sólo un estado. Las flechas en la descripción de las FSM indican la transición del protocolo de un estado a otro. Puesto que en este caso cada una de las máquinas de la Figura 3.9 sólo tiene un estado, necesariamente una transición es del estado a sí mismo (veremos diagramas más complicados enseguida). El suceso que provoca la transición se muestra encima de la línea horizontal que etiqueta la transición y las acciones que se toman cuando tiene lugar el suceso se indican debajo de la línea horizontal. Cuando no se lleve a cabo ninguna acción al ocurrir un suceso, o cuando no se produzca un suceso y se realice una acción, utilizaremos el símbolo   por debajo o por encima de la horizontal, respectivamente, para indicar de manera explícita la ausencia de una acción o de un suceso. El estado inicial de la máquina FSM está indicado mediante la línea de puntos. Aunque las máquinas de estados finitos de la Figura 3.9 tienen un único estado, las que veremos a continuación tendrán múltiples, por lo que es importante identificar el estado inicial de cada máquina FSM.

El lado emisor de `rdt` simplemente acepta datos de la capa superior a través del suceso `rdt_enviar(datos)`, crea un paquete que contiene los datos (mediante la acción `crear_paquete(datos))` y envía el paquete al canal. En la práctica, el suceso `rdt_enviar(datos)` resultaría de una llamada a procedimiento (por ejemplo, a `rdt_enviar()`) realizada por la aplicación de la capa superior.

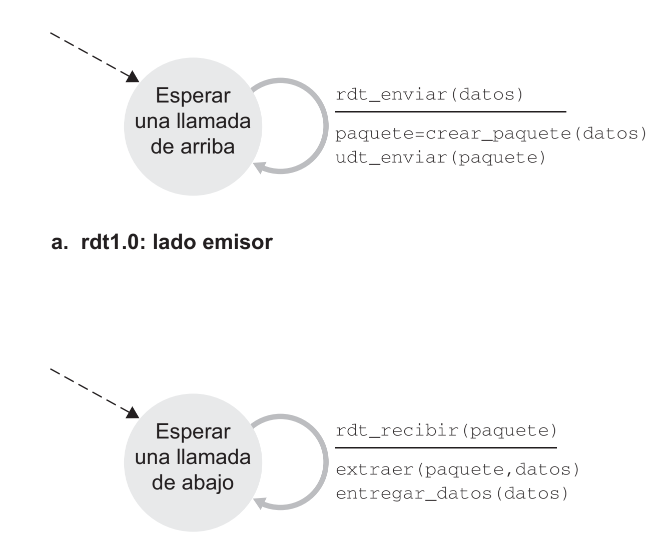


Figura 3.9 • `rdt1.0`: protocolo para un canal totalmente fiable.

En el lado receptor, `rdt` recibe un paquete del canal subyacente a través del suceso `rdt_recibir(paquete)`, extrae los datos del paquete (mediante la acción `extraer(paquete,datos)`) y pasa los datos a la capa superior (mediante la acción `entregar_datos(datos)`). En la práctica, el suceso `rdt_recibir(paquete)` resultaría de una llamada a procedimiento (por ejemplo, a `rdt_recibir()`) desde el protocolo de la capa inferior.

En este protocolo tan simple no existe ninguna diferencia entre una unidad de datos y un paquete. Además, todo el flujo de paquetes va del emisor al receptor, ya que disponiendo de un canal totalmente fiable no existe la necesidad en el lado receptor de proporcionar ninguna realimentación al emisor, puesto que no hay nada que pueda ser incorrecto. Observe que también hemos supuesto que el receptor podía recibir los datos tan rápido como el emisor los enviara. Luego tampoco existe la necesidad de que el receptor le pida al emisor que vaya más despacio.

Transferencia de datos fiable sobre un canal con errores de bit: `rdt2.0`

Un modelo más realista del canal subyacente sería uno en el que los bits de un paquete pudieran corromperse. Normalmente, tales errores de bit se producen en los componentes físicos de una red cuando un paquete se transmite, se propaga o accede a un buffer. Por el momento vamos a continuar suponiendo que todos los paquetes transmitidos son recibidos (aunque sus bits pueden estar corrompidos) en el orden en que se enviaron.

Antes de desarrollar un protocolo que permita una comunicación fiable a través de un canal así, vamos a ver cómo las personas se enfrentan a esta situación. Imagine cómo dictaría un mensaje largo a través del teléfono. En un escenario típico, la persona que escucha el mensaje podría decir "De acuerdo" después de cada frase que escuche, comprenda y apunte. Si la persona que escucha el mensaje no oye una frase, le pedirá que la repita. Este protocolo de dictado de mensajes utiliza tanto **reconocimientos positivos** ("De acuerdo") como **reconocimientos negativos** ("Por favor, repita"). Estos mensajes de control permiten al

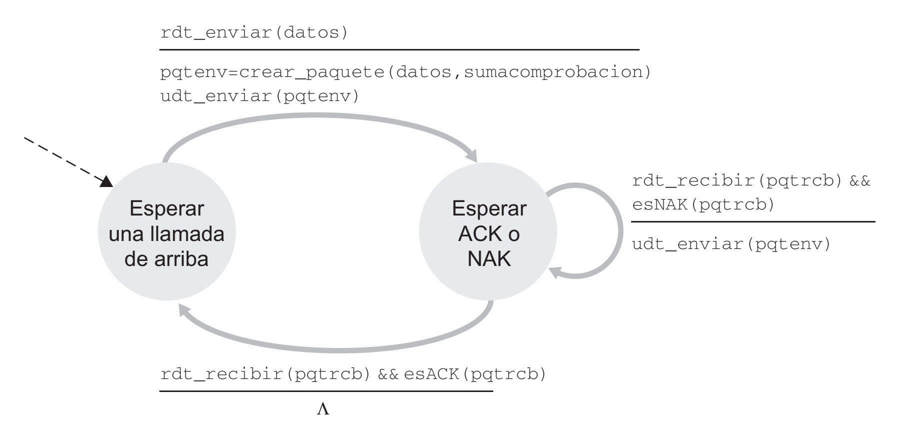
receptor hacer saber al emisor qué es lo que ha recibido correctamente y qué ha recibido con errores y por tanto debe repetir. En una red de computadoras, los protocolos de transferencia de datos fiables basados en tales retransmisiones se conocen como **protocolos ARQ** (***Automatic Repeat reQuest*, Solicitud automática de repetición**).

En los protocolos ARQ se requieren, fundamentalmente, tres capacidades de protocolo adicionales para gestionar la presencia de errores de bit:

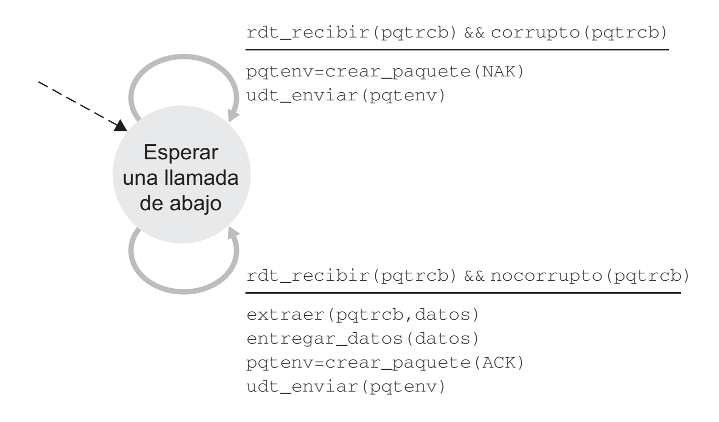
- *Detección de errores*. En primer lugar, se necesita un mecanismo que permita al receptor detectar que se han producido errores de bit. Recuerde de la sección anterior que UDP utiliza el campo de suma de comprobación de Internet precisamente para este propósito. En el Capítulo 5, examinaremos técnicas de detección y corrección de errores con más detalle; estas técnicas permiten al receptor detectar y, posiblemente, corregir los errores de bit en los paquetes. Por el momento, sólo necesitamos saber que estas técnicas requieren que el emisor envíe al receptor bits adicionales (junto con los bits de los datos originales que se desean transferir) y dichos bits también se tendrán en cuenta para el cálculo de la suma de comprobación del paquete de datos `rdt2.0`.
- *Realimentación del receptor*. Dado que el emisor y el receptor normalmente se ejecutan en sistemas terminales diferentes, posiblemente separados por miles de kilómetros, la única forma de que el emisor sepa lo que ocurre en el receptor (en este caso, si un paquete ha sido recibido correctamente o no) es que el receptor envíe explícitamente información de realimentación al emisor. Las respuestas de acuse de recibo o reconocimiento positivo (ACK) y reconocimiento negativo (NAK) en el escenario del dictado de mensajes son ejemplos de esa realimentación. De forma similar, nuestro protocolo `rdt2.0` enviará paquetes ACK y NAK de vuelta desde el receptor al emisor. En principio, estos paquetes sólo necesitan tener una longitud de un bit; por ejemplo, un valor 0 indicaría un reconocimiento negativo (NAK) y un valor 1 indicaría un reconocimiento positivo (ACK).
- *Retransmisión*. Una paquete que se recibe con errores en el receptor será retransmitido por el emisor.

La Figura 3.10 muestra la representación de la máquina de estados finitos para `rdt2.0`, un protocolo de transferencia de datos que dispone de mecanismos de detección de errores y paquetes de reconocimiento positivo y negativo.

El lado emisor de `rdt2.0` tiene dos estados. En el estado más a la izquierda, el protocolo del lado emisor está a la espera de datos procedentes de la capa superior. Cuando se produce el suceso `rdt_enviar(datos)`, el emisor creará un paquete (`pqtenv`) conteniendo los datos que van a ser transmitidos, junto con una suma de comprobación del paquete (como se ha visto en la Sección 3.3.2, por ejemplo, para el caso de un segmento UDP), y luego el paquete se envía mediante la operación `udt_enviar(pqtenv)`. En el estado más a la derecha, el protocolo del emisor está a la espera de un paquete de reconocimiento positivo ACK o negativo NAK procedente del receptor. Si se recibe un paquete ACK (la notación `rdt_recibir(pqtrcb) && esACK (pqtrcb)` mostrada en la Figura 3.10 corresponde a este suceso), el emisor sabe que el paquete más recientemente transmitido ha sido recibido correctamente y, por tanto, el protocolo vuelve al estado de espera de datos procedentes de la capa superior. Si se recibe un reconocimiento negativo NAK, el protocolo retransmite el último paquete y espera a recibir un mensaje ACK o NAK del receptor en respuesta al paquete de datos retransmitido. Es importante observar que cuando el emisor está en el estado “Esperar ACK o NAK”, *no puede* obtener más datos de la capa superior; es



a. rdt2.0: lado emisor



b. rdt2.0: lado receptor

Figura 3.10 • `rdt2.0`: protocolo para un canal con errores de bit.

decir, el suceso `rdt_enviar()` no puede ocurrir; esto sólo ocurrirá después de que el emisor reciba un ACK y salga de ese estado. Por tanto, el emisor no enviará ningún nuevo fragmento de datos hasta estar seguro de que el receptor ha recibido correctamente el paquete actual. Debido a este comportamiento, los protocolos como `rdt2.0` se conocen como protocolos de **parada y espera** (***stop-and-wait protocol***).

El lado receptor de la máquina de estados finitos para `rdt2.0` tiene un único estado. Cuando llega un paquete, el receptor responde con un reconocimiento positivo ACK o negativo NAK, dependiendo de si el paquete recibido es correcto o está corrompido. En la Figura 3.10, la notación `rdt_recibir(pqtrcb) && corrupto(pqtrcb)` corresponde al suceso en el que se ha recibido un paquete y resulta ser erróneo.

Puede parecer que el protocolo `rdt2.0` funciona pero, lamentablemente, tiene un defecto fatal. ¡No hemos tenido en cuenta la posibilidad de que el paquete ACK o NAK pueda estar corrompido! (Antes de continuar, debería pararse a pensar en cómo se podría resolver este problema.) Lamentablemente, este descuido no es tan inocuo como puede parecer. Como mínimo, tendremos que añadir bits de suma de comprobación a los paquetes

ACK/NAK para detectar tales errores. La cuestión más complicada es cómo puede recuperarse el protocolo de los errores en los paquetes ACK o NAK. La dificultad está en que si un paquete ACK o NAK está corrompido, el emisor no tiene forma de saber si el receptor ha recibido o no correctamente el último fragmento de datos transmitido.

Consideremos ahora tres posibilidades para gestionar los paquetes ACK o NAK corruptos:

- Para abordar la primera posibilidad, veamos lo que podría hacer una persona en el escenario del dictado de mensajes. Si la persona que está dictando el mensaje no entiende la respuesta "De acuerdo" o "Por favor, repita" del receptor, probablemente diría "¿Cómo dice?" (lo que introduce un nuevo tipo de paquete del emisor al receptor en nuestro protocolo). A continuación, el receptor repetiría la respuesta. ¿Pero qué ocurriría si el "Cómo dice" está corrompido? El receptor no tendría ni idea de si esa frase formaba parte del dictado o era una solicitud de que repitiera la última respuesta, por lo que probablemente respondería con un "¿Cómo dice usted?". Y, a su vez, por supuesto, dicha respuesta también podría verse alterada. Evidentemente, el problema se complica.
- Una segunda alternativa consistiría en añadir los suficientes bits de suma de comprobación como para permitir al emisor no sólo detectar, sino también recuperarse de los errores de bit. De este modo se resuelve el problema inmediato de un canal que puede corromper los paquetes de datos, pero no perderlos.
- Un tercer método consistiría simplemente en que el emisor reenviara el paquete de datos actual al recibir un paquete ACK o NAK alterado. Sin embargo, este método introduce **paquetes duplicados** en el canal emisor-receptor. La principal dificultad con los paquetes duplicados es que el receptor no sabe si el último paquete ACK o NAK enviado fue recibido correctamente en el emisor. Por tanto, *a priori*, no puede saber si un paquete entrante contiene datos nuevos o se trata de una retransmisión.

Una solución sencilla a este nuevo problema (y que ha sido adoptada en prácticamente todos los protocolos de transferencia de datos existentes, incluido TCP) consiste en añadir un nuevo campo al paquete de datos, y hacer que el emisor numere sus paquetes de datos colocando un **número de secuencia** en este campo. Entonces bastará con que el receptor compruebe ese número de secuencia para determinar si el paquete recibido es o no una retransmisión. Para el caso de este protocolo de parada y espera simple, un número de secuencia de 1 bit será suficiente, ya que le permitirá al receptor saber si el emisor está retransmitiendo el paquete previamente transmitido (el número de secuencia del paquete recibido tiene el mismo número de secuencia que el paquete recibido más recientemente) o si se trata de un paquete nuevo (el número de secuencia es distinto, está desplazado "hacia adelante" en aritmética de módulo 2). Puesto que estamos suponiendo que disponemos de un canal que no pierde datos, los paquetes ACK y NAK no tienen que indicar el número de secuencia del paquete que están confirmando. El emisor sabe que un paquete ACK o NAK recibido (esté alterado o no) fue generado en respuesta a su paquete de datos transmitido más recientemente.

Las Figuras 3.11 y 3.12 muestran la descripción de la máquina de estados finitos para `rdt2.1`, nuestra versión revisada de `rdt2.0`. Ahora las máquinas de estados finitos de los lados emisor y receptor de `rdt2.1` tienen el doble de estados que antes. Esto se debe a que ahora el estado del protocolo tiene que reflejar si el paquete que está enviando actualmente el emisor o el que está esperando el receptor tiene que incluir un número de secuencia igual

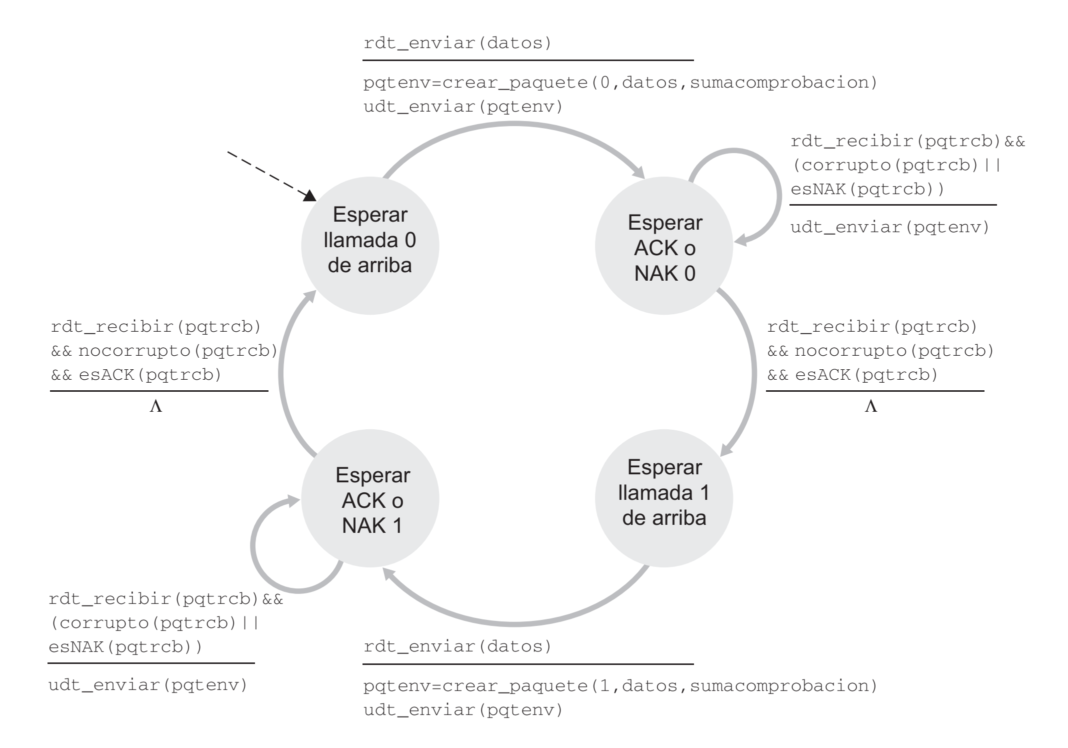


Figura 3.11 • Lado emisor de `rdt2.1`.

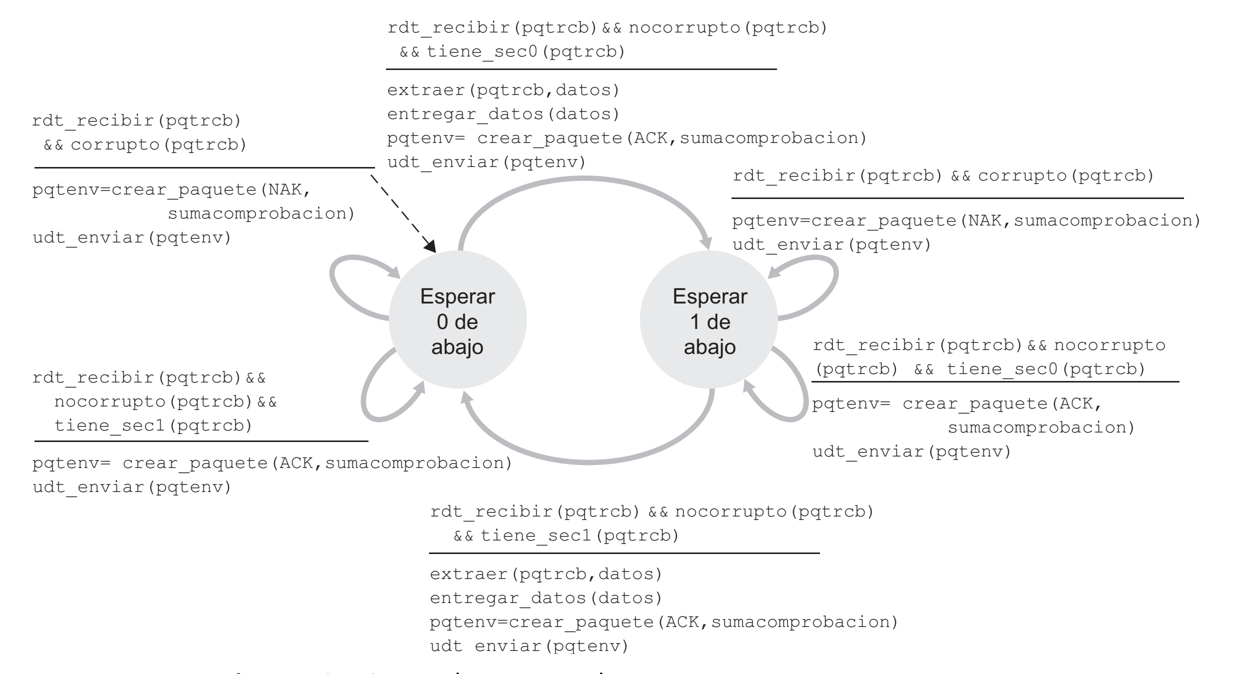


Figura 3.12 • Lado receptor de `rdt2.1`.

a 0 o a 1. Observe que las acciones en aquellos casos en los que un paquete con un número de secuencia de 0 está siendo enviado o es esperado son imágenes especulares de aquellos casos en los que el número de secuencia del paquete es 1; las únicas diferencias se encuentran en la gestión del número de secuencia.

En el protocolo `rdt2.1`, el receptor envía tanto respuestas de reconocimiento positivo como negativo al emisor. Cuando recibe un paquete fuera de secuencia, el receptor envía un paquete ACK para el paquete que ha recibido. Cuando recibe un paquete corrompido, el receptor envía una respuesta de reconocimiento negativo. Podemos conseguir el mismo efecto que con una respuesta NAK si, en lugar de enviar una NAK, enviamos una respuesta de reconocimiento positivo (ACK) para el último paquete recibido correctamente. Un emisor que recibe dos respuestas ACK para el mismo paquete (es decir, recibe respuestas **ACK duplicadas**) sabe que el receptor no ha recibido correctamente el paquete que sigue al que está siendo reconocido (respuesta ACK) dos veces. Nuestro protocolo de transferencia de datos fiable sin respuestas de tipo NAK para un canal con errores de bit es `rdt2.2`, el cual se ilustra en las Figuras 3.13 y 3.14. Una sutil variación entre los protocolos `rtdt2.1` y `rdt2.2` es que ahora el receptor tiene que incluir el número de secuencia del paquete que está siendo confirmado mediante un mensaje ACK (lo que hace incluyendo el argumento `ACK,0` o `ACK,1` en `crear_paquete()` en la máquina de estados finitos de recepción), y el emisor tiene que comprobar el número de secuencia del paquete que está siendo confirmado por el mensaje ACK recibido (lo que se hace incluyendo el argumento `0` o `1` en `esACK()` en la máquina de estados finitos del emisor).

Transferencia de datos fiable sobre un canal con pérdidas y errores de bit: `rdt3.0`

Suponga ahora que además de bits corrompidos, el canal subyacente también puede *perder* paquetes, un suceso no desconocido en las redes de computadoras de hoy en día (incluyendo Internet). Por tanto, ahora el protocolo tiene que preocuparse por dos problemas más: cómo detectar la pérdida de paquetes y qué hacer cuando se pierde un paquete. El uso de la suma de comprobación (*cheksum*), los números de secuencia, los paquetes ACK y la retransmisión de paquetes, técnicas que ya hemos desarrollado en el protocolo `rdt2.2`, nos van a permitir abordar este último problema, Para tratar el primero será necesario añadir un nuevo mecanismo de protocolo.

Hay disponibles muchas formas de abordar la pérdida de paquetes (varias de ellas se exploran en los ejercicios del final del capítulo). Veamos cómo el emisor puede detectar la pérdida de paquetes y cómo puede recuperarse de la misma. Suponga que el emisor transmite un paquete de datos y bien el propio paquete o el mensaje ACK del receptor para ese paquete se pierde. En cualquiera de estos dos casos, al emisor no le llega ninguna respuesta procedente del receptor. Si el emisor está dispuesto a esperar el tiempo suficiente como para estar *seguro* de que se ha perdido un paquete, simplemente puede retransmitirlo. Se puede comprobar de un modo sencillo que efectivamente este protocolo funciona.

Pero, ¿cuánto tiempo tiene que esperar el emisor para estar seguro de que se ha perdido un paquete? Es evidente que el emisor tiene que esperar al menos un tiempo igual al retardo de ida y vuelta entre el emisor y el receptor (lo que puede incluir el almacenamiento temporal en los buffers de los routers intermedios) más una cierta cantidad de tiempo que será necesaria para procesar un paquete en el receptor. En muchas redes, este retardo máximo del caso peor es muy difícil incluso de estimar y aún más de conocer con precisión. Además, idealmente, el protocolo debería recuperarse de la pérdida de paquetes tan

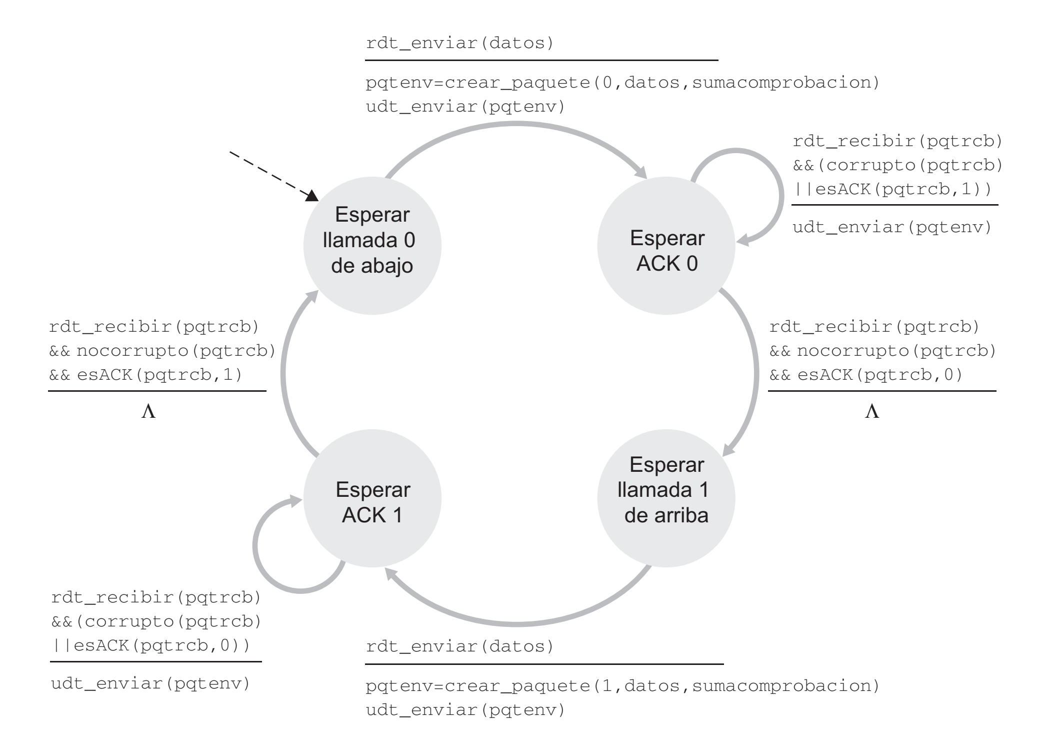


Figura 3.13 • Lado emisor de `rdt2.2`.

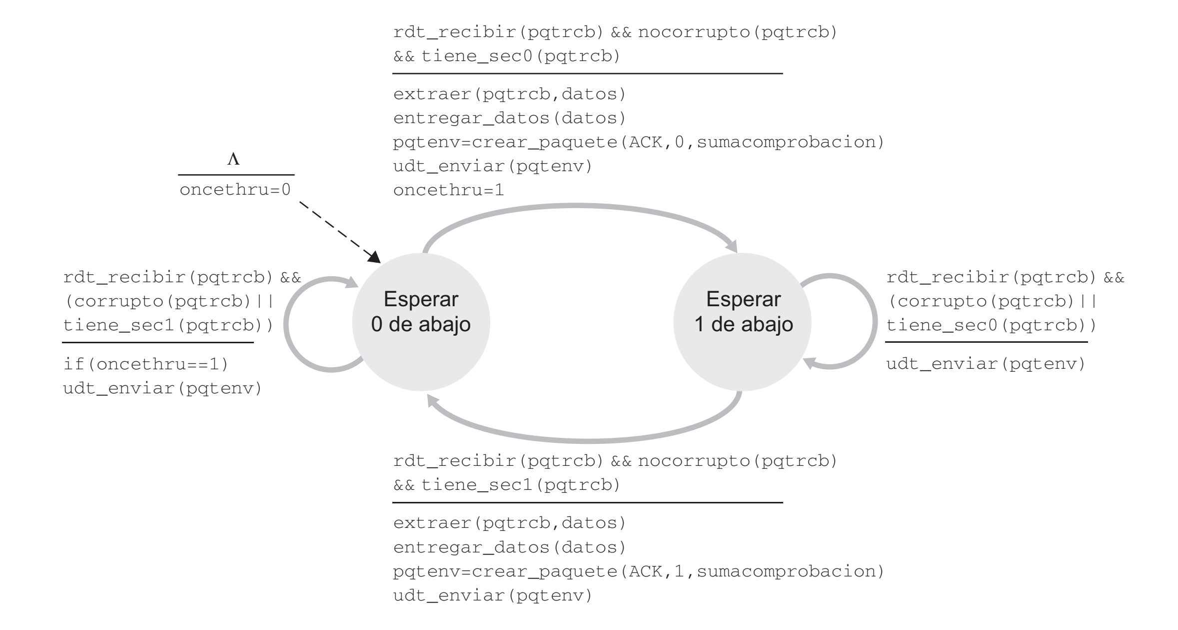


Figura 3.14 • Lado receptor de `rdt2.2`.

pronto como fuera posible; pero si espera un tiempo igual al retardo en el caso peor, eso significa una larga espera hasta iniciar el mecanismo de recuperación de errores. Por tanto, el método que se adopta en la práctica es que el emisor seleccione juiciosamente un intervalo de tiempo tal que sea probable que un paquete se haya perdido, aunque no sea seguro que tal pérdida se haya producido. Si dentro de ese intervalo de tiempo no se ha recibido un ACK, el paquete se retransmite. Observe que si un paquete experimenta un retardo particularmente grande, el emisor puede retransmitirlo incluso aunque ni el paquete de datos ni su correspondiente ACK se hayan perdido. Esto introduce la posibilidad de que existan **paquetes de datos duplicados** en el canal emisor-receptor. Afortunadamente, el protocolo `rdt2.2` ya dispone de la funcionalidad (los números de secuencia) para afrontar la existencia de paquetes duplicados.

Desde el punto de vista del emisor, la retransmisión es la solución para todo. El emisor no sabe si se ha perdido un paquete de datos, se ha perdido un mensaje ACK, o simplemente el paquete o el ACK están retardados. En todos los casos, la acción es la misma: retransmitir. La implementación de un mecanismo de retransmisión basado en el tiempo requiere un **temporizador de cuenta atrás** que pueda interrumpir al emisor después de que haya transcurrido un determinado periodo de tiempo. Por tanto, el emisor necesitará poder (1) iniciar el temporizador cada vez que envíe un paquete (bien por primera vez o en una retransmisión), (2) responder a una interrupción del temporizador (ejecutando las acciones apropiadas) y (3) detener el temporizador.

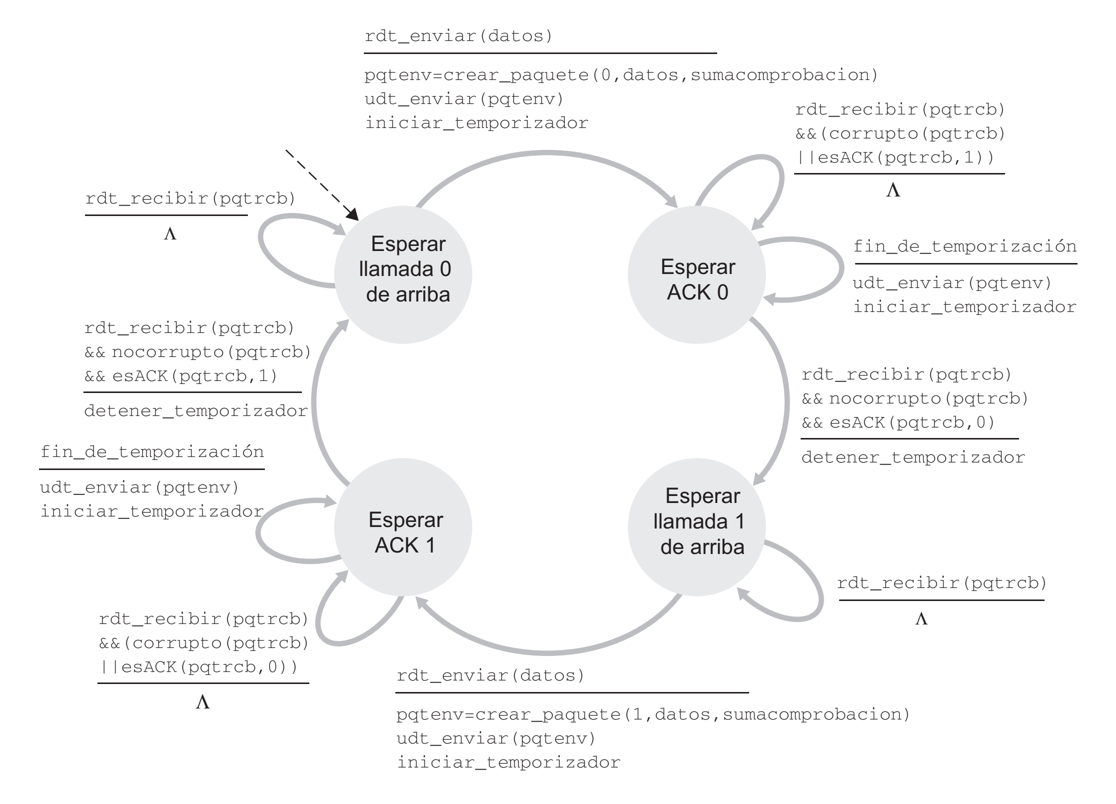


Figura 3.15 • Lado emisor de `rdt3.0`.

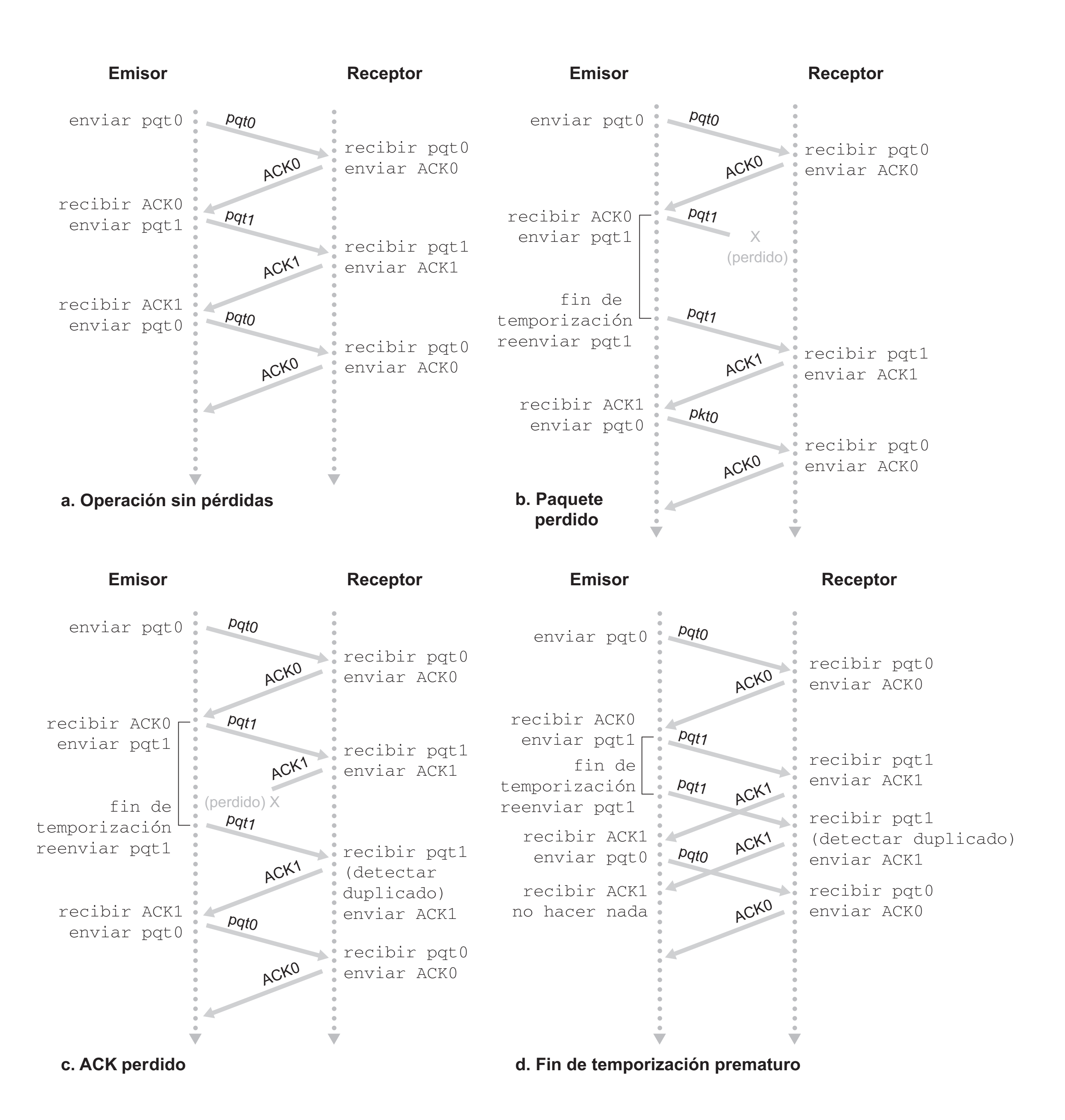


Figura 3.16 • Operación de `rdt3.0`, el protocolo de bit alternante.

La Figura 3.15 muestra la máquina de estados finitos del emisor para `rdt3.0`, un protocolo que transfiere datos de forma fiable a través de un canal que puede corromper o perder paquetes; en los problemas de repaso, se le pedirá que defina la máquina de estados finitos del receptor para `rdt3.0`. La Figura 3.16 muestra cómo opera el protocolo cuando

no se pierden ni se retardan los paquetes y cómo gestiona la pérdida de paquetes de datos. En la Figura 3.16, el tiempo va avanzando desde la parte superior del diagrama hacia la parte inferior del mismo; observe que el instante de recepción de un paquete es necesariamente posterior al instante de envío de un paquete como consecuencia de los retardos de transmisión y de propagación. En las Figuras 3.16(b)–(d), los corchetes en el lado del emisor indican los instantes de inicio y fin del temporizador. Algunos de los aspectos más sutiles de este protocolo se verán en los ejercicios incluidos al final del capítulo. Dado que los números de secuencia de los paquetes alternan entre 0 y 1, el protocolo `rdt3.0` se denomina en ocasiones **protocolo de bit alternante**.

Hasta aquí hemos ensamblado los elementos clave de un protocolo de transferencia de datos. Las sumas de comprobación, los números de secuencia, los temporizadores y los paquetes de reconocimiento positivo y negativo desempeñan un papel fundamental y necesario en el funcionamiento del protocolo. A continuación vamos a trabajar con un protocolo de transferencia de datos fiable.

3.4.2 Protocolo de transferencia de datos fiable con procesamiento en cadena

El protocolo `rdt3.0` es un protocolo funcionalmente correcto, pero es muy improbable que haya alguien a quien satisfaga su rendimiento, especialmente en las redes de ata velocidad actuales. La base del problema del rendimiento de `rdt3.0` se encuentra en el hecho de que es un protocolo de parada y espera.

Para entender el impacto sobre el rendimiento de este comportamiento de parada y espera, vamos a considerar un caso ideal de dos hosts, uno localizado en la costa oeste de Estados Unidos y el otro en la costa este, como se muestra en la Figura 3.17. El retardo de propagación de ida y vuelta, RTT, a la velocidad de la luz entre estos dos sistemas terminales es aproximadamente igual a 30 milisegundos. Suponga que están conectados mediante un canal cuya velocidad de transmisión, *R*, es de 1 Gbps (10^9 bits por segundo). Con un tamaño de paquete, *L*, de 1.000 bytes (8.000 bits) por paquete, incluyendo los campos de cabecera y los datos, el tiempo necesario para transmitir el paquete por un enlace de 1 Gbps es:

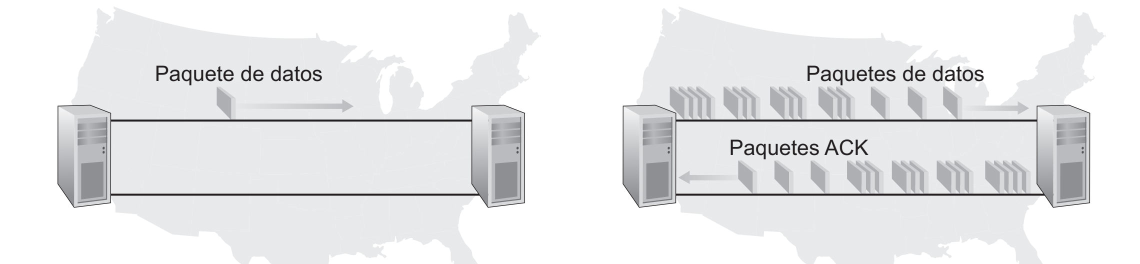
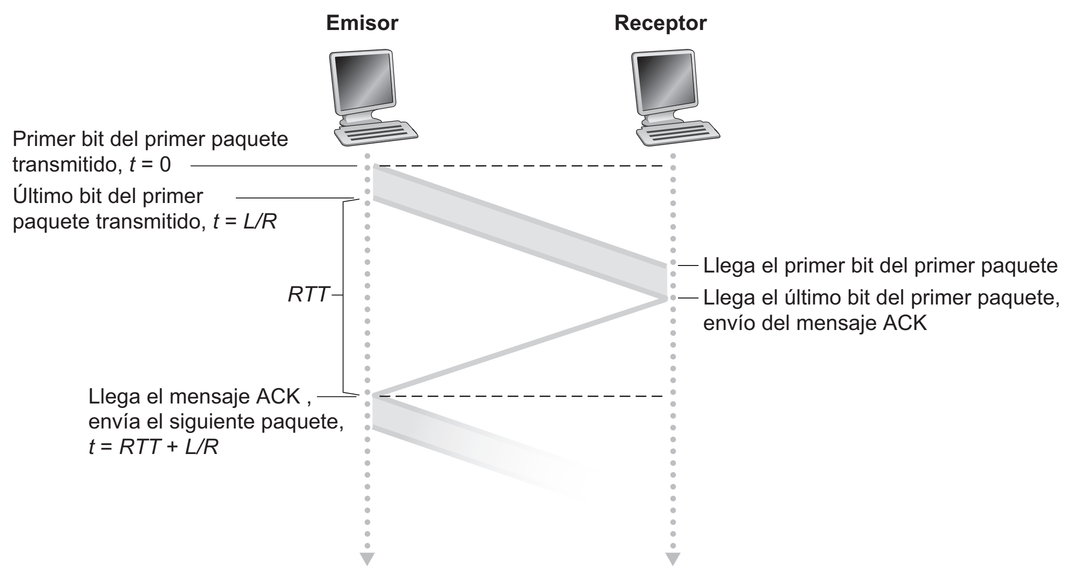
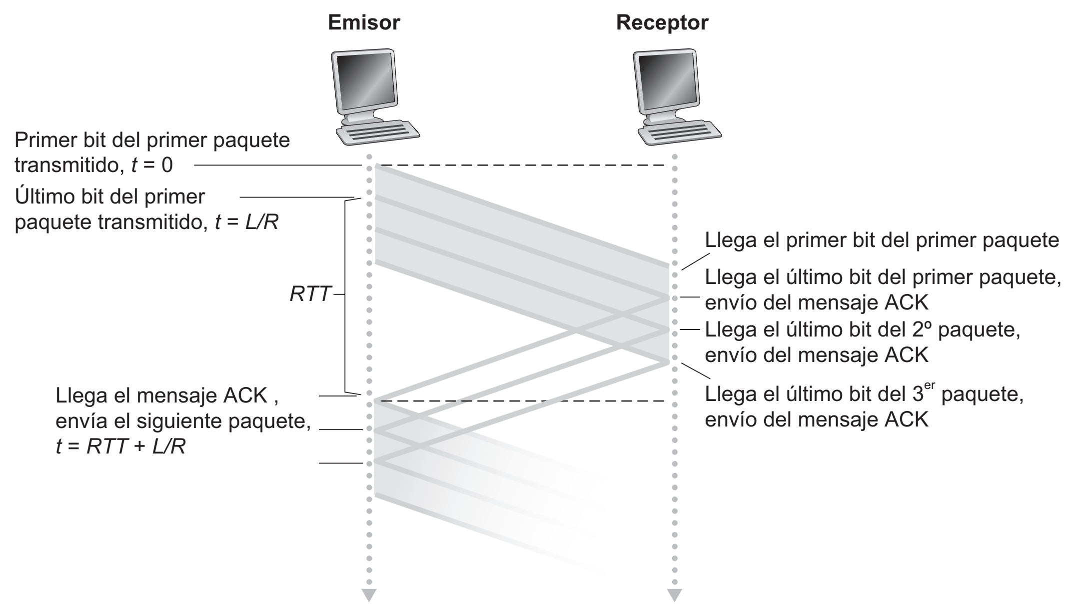


Figura 3.17 • Protocolo de parada y espera y protocolo con procesamiento en cadena.

$$d_{trans} = \frac{L}{R} = \frac{8000 \text{ bits/paquete}}{10^9 \text{ bits/segundo}} = 8 \text{ microsegundos}$$



a. Funcionamiento de un protocolo de parada y espera.



b. Funcionamiento con procesamiento en cadena.

Figura 3.18 • Proceso de transmisión con un protocolo de parada y espera y un protocolo con procesamiento en cadena.

La Figura 3.18(a) muestra que, con nuestro protocolo de parada y espera, si el emisor comienza a transmitir el paquete en el instante $t = 0$, entonces en el instante $t = L/R = 8$ microsegundos el último bit entra en el canal en el lado del emisor. El paquete entonces tarda 15 milisegundos en atravesar el país, emergiendo el último bit del paquete en el lado del receptor en el instante $t = RTT/2 + L/R = 15{,}008$ milisegundos. Con el fin de simplificar, vamos a suponer que los paquetes de reconocimiento ACK son extremadamente pequeños (por lo que podemos ignorar su tiempo de transmisión) y que el receptor puede enviar un ACK tan pronto como ha recibido el último bit de un paquete de datos, llegando dicho ACK al emisor en $t = RTT + L/R = 30{,}008$ mseg. En esta situación, ahora el emisor puede transmitir el siguiente mensaje. Por tanto, en 30,008 milisegundos, el emisor ha estado transmitiendo durante sólo 0,008 milisegundos. Si definimos la tasa de utilización del emisor (o del canal) como la fracción de tiempo que el emisor está realmente ocupado enviando bits al canal, el análisis de la Figura 3.18(a) muestra que el protocolo de parada y espera tiene una tasa de utilización del emisor, U_{emisor}, bastante mala de

$$U_{\text{emisor}} = \frac{L/R}{RTT + L/R} = \frac{0{,}008}{30{,}008} = 0{,}00027$$

Es decir, ¡el emisor sólo está ocupado 2,7 diezmilésimas del tiempo! En otras palabras, el emisor sólo ha podido enviar 1.000 bytes en 30,008 milisegundos, una tasa de transferencia efectiva de sólo 267 kbps, incluso disponiendo de un enlace a 1 Gbps. Imagine al infeliz administrador de la red que ha pagado una fortuna por un enlace con una capacidad de un gigabit para obtener una tasa de transferencia de únicamente 267 kilobits por segundo. Éste es un ejemplo gráfico de cómo los protocolos de red pueden limitar las capacidades proporcionadas por el hardware de red subyacente. Además, hemos despreciado los tiempos de procesamiento del protocolo de la capa inferior tanto en el emisor como en el receptor, así como los retardos de procesamiento y de cola que pueden tener lugar en cualquiera de los routers intermedios existentes entre el emisor y el receptor. La inclusión de estos efectos sólo serviría para incrementar el retardo y acentuar más su pésimo rendimiento.

La solución a este problema de rendimiento concreto es simple: en lugar de operar en el modo parada y espera, el emisor podría enviar varios paquetes sin esperar a los mensajes de reconocimiento, como se ilustra en la Figura 3.17(b). La Figura 3.18(b) muestra que si el emisor transmite tres paquetes antes de tener que esperar a los paquetes de reconocimiento, la utilización del emisor prácticamente se triplica. Dado que los muchos paquetes que están en tránsito entre el emisor y el receptor pueden visualizarse como el relleno de un conducto (*pipeline*), esta técnica se conoce como ***pipelining*** o **procesamiento en cadena**. El procesamiento en cadena tiene las siguientes consecuencias en los protocolos de transferencia de datos fiables:

- El rango de los números de secuencia tiene que incrementarse, dado que cada paquete en tránsito (sin contar las retransmisiones) tiene que tener un número de secuencia único y pueden coexistir múltiples paquetes en tránsito que no hayan sido confirmados mediante un reconocimiento.
- Los lados emisor y receptor de los protocolos pueden tener que almacenar en buffer más de un paquete. Como mínimo, el emisor tendrá en el buffer los paquetes que han sido transmitidos pero que todavía no han sido reconocidos. Como veremos enseguida, también puede ser necesario almacenar en el buffer del receptor los paquetes recibidos correctamente.

- El rango necesario de los números de secuencia y los requisitos de buffer dependerán de la forma en que un protocolo de transferencia de datos responda a la pérdida de paquetes y a los paquetes corrompidos o excesivamente retardados. Hay disponibles dos métodos básicos que permiten la recuperación de errores mediante procesamiento en cadena: **Retroceder N** y la **repetición selectiva**.

3.4.3 Retroceder N (GBN)

En un **protocolo GBN (*Go-Back-N,* Retroceder N)**, el emisor puede transmitir varios paquetes (si están disponibles) sin tener que esperar a que sean reconocidos, pero está restringido a no tener más de un número máximo permitido, *N,* de paquetes no reconocidos en el canal. En esta sección vamos a describir el protocolo GBN con cierto detalle. Pero antes de seguir leyendo, le animamos a practicar con el applet GBN (¡un applet impresionante!) disponible en el sitio web del libro.

La Figura 3.19 muestra la parte correspondiente al emisor del rango de los números de secuencia en un protocolo GBN. Si definimos la `base` como el número de secuencia del paquete no reconocido más antiguo y `signumsec` como el número de secuencia más pequeño no utilizado (es decir, el número de secuencia del siguiente paquete que se va a enviar), entonces se pueden identificar los cuatro intervalos en rango de los números de secuencia. Los números de secuencia pertenecientes al intervalo `[0,base-1]` corresponden a paquetes que ya han sido transmitidos y reconocidos. El intervalo `[base,signumsec-1]` corresponde a paquetes que ya han sido enviados pero todavía no se han reconocido. Los números de secuencia del intervalo `[signumsec,base+N-1]` se pueden emplear para los paquetes que pueden ser enviados de forma inmediata, en caso de que lleguen datos procedentes de la capa superior. Y, por último, los números de secuencia mayores o iguales que `base+N` no pueden ser utilizados hasta que un paquete no reconocido que se encuentre actualmente en el canal sea reconocido (específicamente, el paquete cuyo número de secuencia sea igual a `base`).

Como sugiere la Figura 3.19, el rango de los números de secuencia permitidos para los paquetes transmitidos pero todavía no reconocidos puede visualizarse como una ventana de tamaño *N* sobre el rango de los números de secuencia. Cuando el protocolo opera, esta ventana se desplaza hacia adelante sobre el espacio de los números de secuencia. Por esta razón, *N* suele denominarse **tamaño de ventana** y el propio protocolo GBN se dice que es un **protocolo de ventana deslizante**. Es posible que se esté preguntando, para empezar, por qué debemos limitar a *N* el número de paquetes no reconocidos en circulación. ¿Por qué no permitir un número ilimitado de tales paquetes? En la Sección 3.5 veremos que el control de

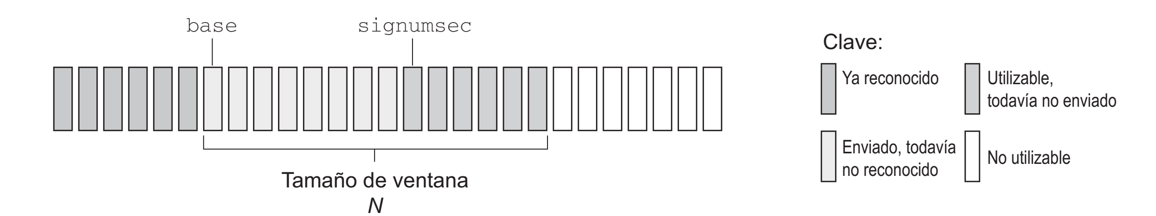


Figure 3.19 • Números de secuencia en el emisor en el protocolo Retroceder N.

flujo es una de las razones para imponer un límite en el emisor. En la Sección 3.7 examinaremos otra razón al estudiar el mecanismo de control de congestión de TCP.

En la práctica, el número de secuencia de un paquete se incluye en un campo de longitud fija de la cabecera del paquete. Si k es el número de bits contenido en el campo que especifica el número de secuencia del paquete, el rango de los números de secuencia será $[0,2^k - 1]$. Con un rango finito de números de secuencia, todas las operaciones aritméticas que impliquen a los números de secuencia tendrán que efectuarse utilizando aritmética en módulo 2^k. (Es decir, el espacio de números de secuencia puede interpretarse como un anillo de tamaño 2^k, donde el número de secuencia $2^k - 1$ va seguido por el número de secuencia 0.) Recuerde que el protocolo `rdt3.0` dispone de un número de secuencia de 1 bit y de un rango de números de secuencia de [0,1]. Algunos de los problemas incluidos al final del capítulo exploran las consecuencias de disponer de un rango finito de números de secuencia. En la Sección 3.5 veremos que TCP utiliza un campo para el número de secuencia de 32 bits, donde los números de secuencia cuentan los bytes del flujo de datos, en lugar de los paquetes.

Las Figuras 3.20 y 3.21 proporcionan una descripción ampliada de la máquina de estados finitos de los lados emisor y receptor de un protocolo GBN basado en paquetes de reconocimiento ACK y que no emplea paquetes de reconocimiento NAK. Nos referiremos a esta descripción de una FSM como *FSM ampliada*, ya que hemos añadido variables (similares a las variables de un lenguaje de programación) para `base` y `signumsec`, y operaciones sobre dichas variables y acciones condicionales que las implican. Observe que la especificación de una FSM ampliada empieza a asemejarse a una especificación de un lenguaje de programación. [Bochman 1984] proporciona un excelente repaso de otras ampliaciones de las técnicas de máquinas de estados finitos, así como de técnicas basadas en lenguajes de programación para la especificación de protocolos.

El emisor del protocolo GBN tiene que responder a tres tipos de sucesos:

- *Invocación desde la capa superior.* Cuando se llama a `rdt_enviar()` desde la capa superior, lo primero que hace el emisor es ver si la ventana está llena; es decir, si hay N paquetes no reconocidos en circulación. Si la ventana no está llena, se crea y se envía un paquete y se actualizan las variables de la forma apropiada. Si la ventana está llena, el emisor simplemente devuelve los datos a la capa superior, indicando de forma implícita que la ventana está llena. Probablemente entonces la capa superior volverá a intentarlo más tarde. En una implementación real, muy posiblemente el emisor almacenaría en el buffer estos datos (pero no los enviaría de forma inmediata) o dispondría de un mecanismo de sincronización (por ejemplo, un semáforo o un indicador) que permitiría a la capa superior llamar a `rdt_enviar()` sólo cuando la ventana no estuviera llena.
- *Recepción de un mensaje de reconocimiento ACK.* En nuestro protocolo GBN, un reconocimiento de un paquete con un número de secuencia n implica un **reconocimiento acumulativo**, lo que indica que todos los paquetes con un número mayor o igual que n han sido correctamente recibidos por el receptor. Volveremos sobre este tema enseguida al examinar el lado receptor de GBN.
- *Un suceso de fin de temporización.* El nombre de este protocolo, "Retroceder N", se deriva del comportamiento del emisor en presencia de paquetes perdidos o muy retardados. Como en los protocolos de parada y espera, se empleará un temporizador para recuperarse de la pérdida de paquetes de datos o de reconocimiento de paquetes. Si se produce un fin de temporización, el emisor reenvía *todos* los paquetes que haya transmi-

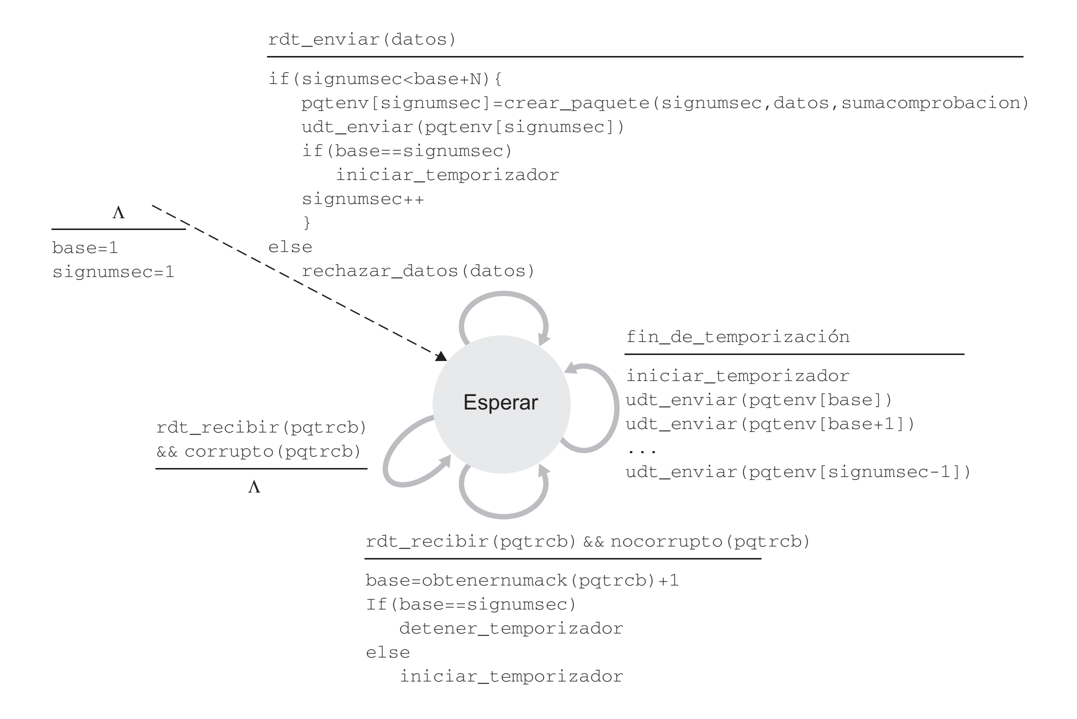


Figura 3.20 • Descripción de la FSM ampliada del lado de emisión de GBN.

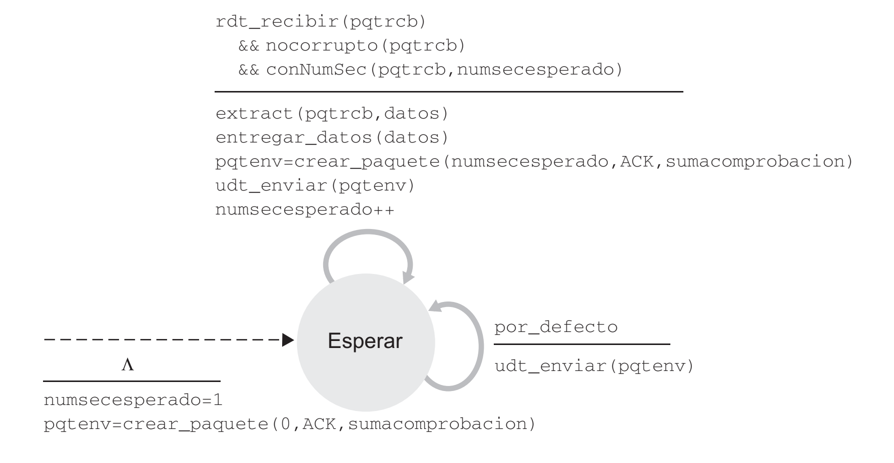


Figura 3.21 • Descripción de la FSM ampliada del lado receptor de GBN.

tido anteriormente y que todavía no hayan sido reconocidos. El emisor de la Figura 3.20 utiliza un único temporizador, que puede interpretarse como un temporizador para los paquetes transmitidos pero todavía no reconocidos. Si se recibe un paquete ACK pero existen más paquetes transmitidos adicionales no reconocidos, entonces se reinicia el temporizador. Si no hay paquetes no reconocidos en circulación, el temporizador se detiene.

Las acciones del receptor en el protocolo GBN también son simples. Si un paquete con un número de secuencia *n* se recibe correctamente y en orden (es decir, los últimos datos entregados a la capa superior proceden de un paquete con el número de secuencia $n - 1$), el receptor envía un paquete ACK para el paquete *n* y entrega la parte de los datos del paquete a la capa superior. En todos los restantes casos, el receptor descarta el paquete y reenvía un mensaje ACK para el paquete recibido en orden más recientemente. Observe que dado que los paquetes se entregan a la capa superior de uno en uno, si el paquete *k* ha sido recibido y entregado, entonces todos los paquetes con un número de secuencia menor que *k* también han sido entregados. Por tanto, el uso de confirmaciones acumulativas es una opción natural del protocolo GBN.

En nuestro protocolo GBN, el receptor descarta los paquetes que no están en orden. Aunque puede parecer algo tonto y una pérdida de tiempo descartar un paquete recibido correctamente (pero desordenado), existe una justificación para hacerlo. Recuerde que el receptor debe entregar los datos en orden a la capa superior. Suponga ahora que se espera el paquete *n*, pero llega el paquete $n + 1$. Puesto que los datos tienen que ser entregados en orden, el receptor *podría* guardar en el buffer el paquete $n + 1$ y luego entregar ese paquete a la capa superior después de haber recibido y entregado el paquete *n*. Sin embargo, si se pierde el paquete *n*, tanto él como el paquete $n + 1$ serán retransmitidos como resultado de la regla de retransmisión del protocolo GBN en el lado de emisión. Por tanto, el receptor puede simplemente descartar el paquete $n + 1$. La ventaja de este método es la simplicidad del almacenamiento en el buffer del receptor (el receptor no necesita almacenar en el buffer *ninguno* de los paquetes entregados desordenados. Por tanto, mientras el emisor tiene que mantener los límites inferior y superior de su ventana y la posición de `signumsec` dentro de esa ventana, el único fragmento de información que el receptor debe mantener es el número de secuencia del siguiente paquete en orden. Este valor se almacena en la variable `numsec-esperado`, como se muestra en la máquina de estados finitos del receptor de la Figura 3.21. Por supuesto, la desventaja de descartar un paquete correctamente recibido es que la subsiguiente retransmisión de dicho paquete puede perderse o alterarse y, por tanto, ser necesarias aún más retransmisiones.

La Figura 3.22 muestra el funcionamiento del protocolo GBN para el caso de un tamaño de ventana de cuatro paquetes. A causa de esta limitación del tamaño de ventana, el emisor transmite los paquetes 0 a 3, pero tiene que esperar a que uno o más de estos paquetes sean reconocidos antes de continuar. A medida que se reciben los sucesivos paquetes ACK (por ejemplo, `ACK0` y `ACK1`), la ventana se desplaza hacia adelante y el emisor puede transmitir un nuevo paquete (`pqt4` y `pqt5`, respectivamente). En el lado receptor se pierde el paquete 2 y, por tanto, los paquetes 3, 4 y 5 no se reciben en el orden correcto y, lógicamente, se descartan.

Antes de terminar esta exposición acerca de GBN, merece la pena destacar que una implementación de este protocolo en una pila de protocolos tendría probablemente una estructura similar a la de la máquina de estados finitos ampliada de la Figura 3.20. Posiblemente, la implementación se realizaría mediante varios procedimientos que implementasen las acciones que habría que realizar en respuesta a los distintos sucesos que pueden producirse. En una **programación basada en sucesos**, los diversos procedimientos son llamados (invocados) por otros procedimientos de la pila de protocolos, o como resultado de una interrupción. En el emisor, estos sucesos podrían ser (1) una llamada de una entidad de la capa superior para invocar `rdt_enviar()`, (2) una interrupción del temporizador y (3) una llamada de la capa inferior para invocar `rdt_recibir()` cuando llega un paquete. Los ejerci-

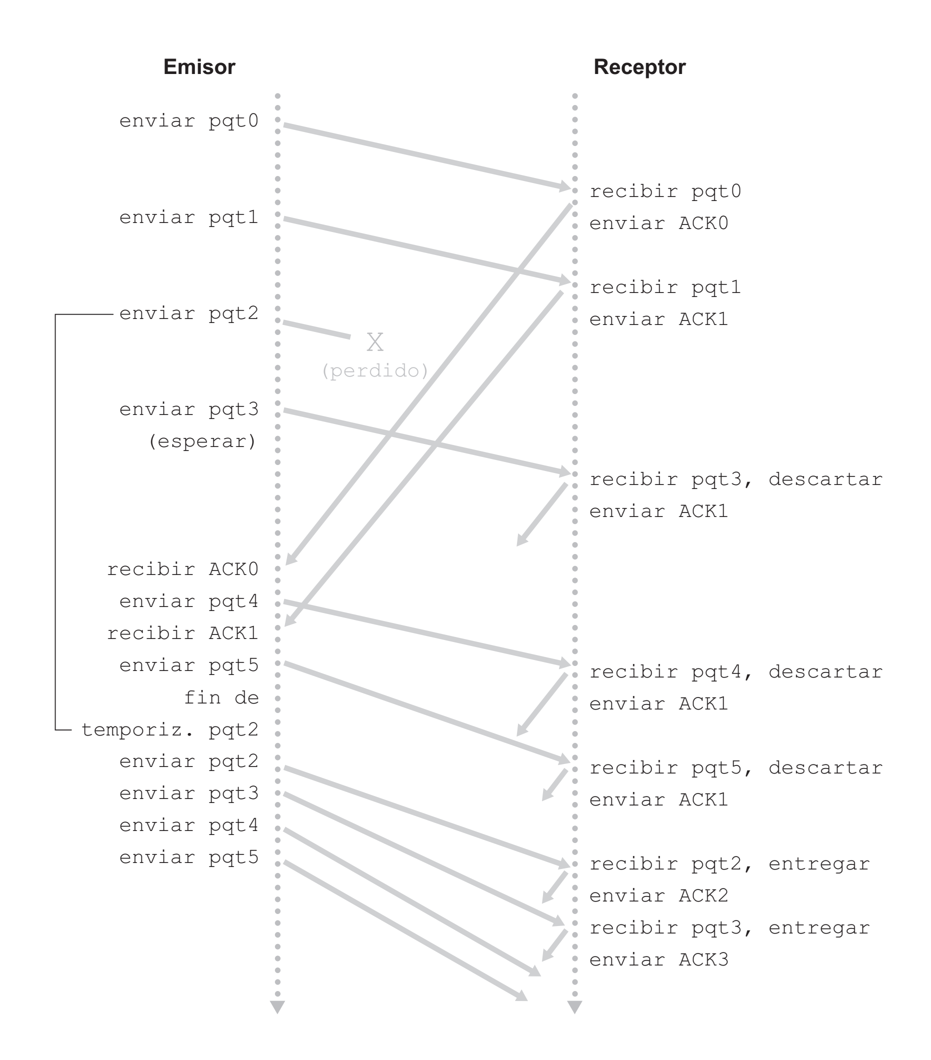


Figura 3.22 • Funcionamiento del protocolo GBN (Retroceder N).

cios sobre programación incluidos al final del capítulo le darán la oportunidad de implementar en la práctica estas rutinas en una red simulada, pero realista.

Observe que el protocolo GBN incorpora casi todas las técnicas que veremos en la Sección 3.5 al estudiar los componentes del servicio de transferencia de datos fiable de TCP. Estas técnicas incluyen el uso de números de secuencia, reconocimientos acumulativos, sumas de comprobación y una operación de fin de temporización/retransmisión.

3.4.4 Repetición selectiva (SR)

El protocolo GBN permite al emisor, en teoría, "llenar el conducto" mostrado en la Figura 3.17 con paquetes, evitando así los problemas de utilización del canal que hemos visto con los protocolos de parada y espera. Sin embargo, hay algunos escenarios en los que el propio GBN presenta problemas de rendimiento. En particular, cuando el tamaño de la ventana y el

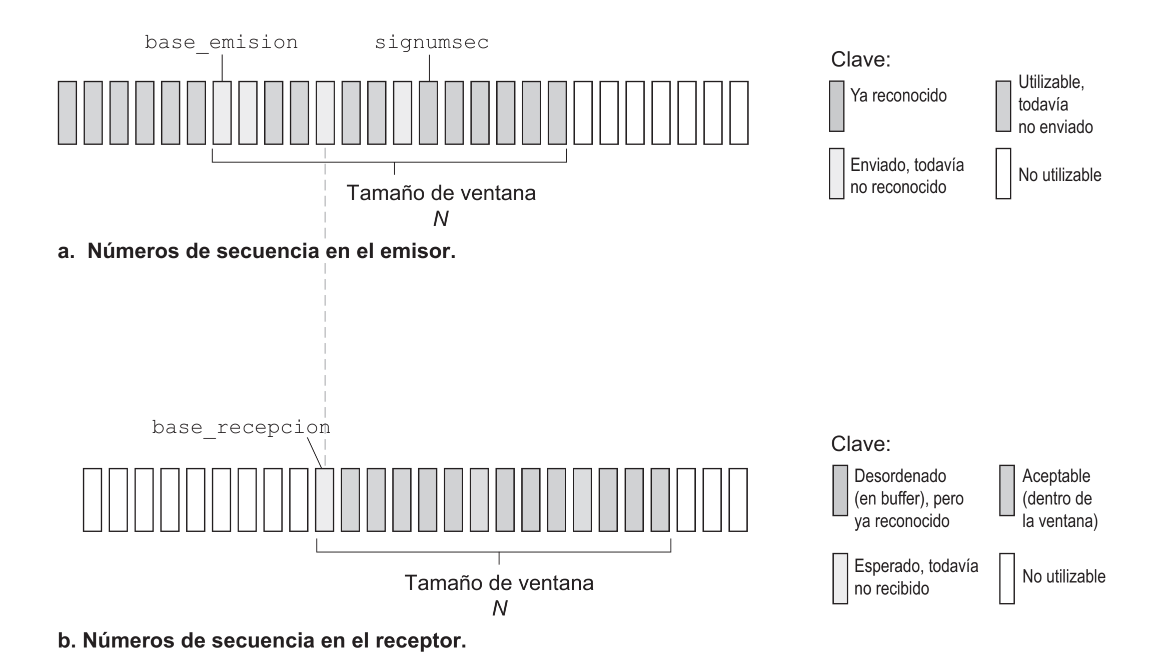


Figura 3.23 • Espacios de números de secuencia del lado emisor y del lado receptor en el protocolo de repetición selectiva (SR).

producto ancho de banda-retardo son grandes puede haber muchos paquetes en el canal. En este caso, un único paquete erróneo podría hacer que el protocolo GBN retransmitiera una gran cantidad de paquetes, muchos de ellos de forma innecesaria. A medida que la probabilidad de errores en el canal aumenta, el canal puede comenzar a llenarse con estas retransmisiones innecesarias. En nuestro escenario del dictado de mensajes, imagine que si cada vez que se altera una palabra, las 1.000 palabras que la rodean (por ejemplo, con un tamaño de ventana de 1.000 palabras) tuvieran que ser repetidas, el dictado se ralentizaría a causa de la repetición de palabras.

Como su nombre sugiere, los protocolos de repetición selectiva evitan las retransmisiones innecesarias haciendo que el emisor únicamente retransmita aquellos paquetes que se sospeche que llegaron al receptor con error (es decir, que se perdieron o estaban corrompidos). Esta retransmisión individualizada y necesaria requerirá que el receptor confirme *individualmente* qué paquetes ha recibido correctamente. De nuevo, utilizaremos una ventana de tamaño *N* para limitar el número de paquetes no reconocidos y en circulación en el canal. Sin embargo, a diferencia de GBN, el emisor ya habrá recibido mensajes ACK para algunos de los paquetes de la ventana. En la Figura 3.23 se muestra el espacio de números de secuencia del lado de emisión del protocolo SR. La Figura 3.24 detalla las distintas acciones que lleva a cabo el emisor del protocolo SR.

El receptor de SR confirmará que un paquete se ha recibido correctamente tanto si se ha recibido en el orden correcto como si no. Los paquetes no recibidos en orden se almacenarán en el buffer hasta que se reciban los paquetes que faltan (es decir, los paquetes con números de secuencia menores), momento en el que un lote de paquetes puede entregarse en orden a la capa superior. En la Figura 3.25 se enumeran las acciones tomadas por el receptor

1. *Datos recibidos de la capa superior.* Cuando se reciben datos de la capa superior, el emisor de SR comprueba el siguiente número de secuencia disponible para el paquete. Si el número de secuencia se encuentra dentro de la ventana del emisor, los datos se empaquetan y se envían; en caso contrario, bien se almacenan en el buffer o bien se devuelven a la capa superior para ser transmitidos más tarde, como en el caso del protocolo GBN.
2. *Fin de temporización.* De nuevo, se emplean temporizadores contra la pérdida de paquetes. Sin embargo, ahora, cada paquete debe tener su propio temporizador lógico, ya que sólo se transmitirá un paquete al producirse el fin de la temporización. Se puede utilizar un mismo temporizador hardware para imitar el funcionamiento de varios temporizadores lógicos [Varghese 1997].
3. *ACK recibido.* Si se ha recibido un mensaje ACK, el emisor de SR marca dicho paquete como que ha sido recibido, siempre que esté dentro de la ventana. Si el número de secuencia del paquete es igual a `base_emision`, se hace avanzar la base de la ventana, situándola en el paquete no reconocido que tenga el número de secuencia más bajo. Si la ventana se desplaza y hay paquetes que no han sido transmitidos con números de secuencia que ahora caen dentro de la ventana, entonces esos paquetes se transmiten.

Figura 3.24 • Sucesos y acciones en el lado emisor del protocolo SR.

1. *Se ha recibido correctamente un paquete cuyo número de secuencia pertenece al intervalo* `[base_recepcion, base_recepcion+N-1]`. En este caso, el paquete recibido cae dentro de la ventana del receptor y se devuelve al emisor un paquete ACK selectivo. Si el paquete no ha sido recibido con anterioridad, se almacena en el buffer. Si este paquete tiene un número de secuencia igual a la base de la ventana de recepción (`base_recepcion` en la Figura 3.22), entonces este paquete y cualquier paquete anteriormente almacenado en el buffer y numerado consecutivamente (comenzando por `base_recepcion`) se entregan a la capa superior. La ventana de recepción avanza entonces el número de paquetes suministrados entregados a la capa superior. Por ejemplo, en la Figura 3.26, cuando se recibe un paquete con el número de secuencia `base_recepcion = 2`, éste y los paquetes 3, 4 y 5 pueden entregarse a la capa superior.
2. *Se ha recibido correctamente un paquete cuyo número de secuencia pertenece al intervalo* `[base_recepcion -N, base_recepcion -1]`. En este caso, se tiene que generar un mensaje ACK, incluso aunque ese paquete haya sido reconocido anteriormente por el receptor.
3. *En cualquier otro caso.* Ignorar el paquete.

Figura 3.25 • Sucesos y acciones en el lado receptor del protocolo SR.

de SR. La Figura 3.26 muestra un ejemplo del funcionamiento de SR en presencia de paquetes perdidos. Observe que, en esta figura, inicialmente el receptor almacena en el buffer los paquetes 3, 4 y 5, y luego los entrega, junto con el paquete 2, a la capa superior una vez que se ha recibido dicho paquete 2.

Es importante observar en el Paso 2 de la Figura 3.25 que el receptor vuelve a reconocer (en lugar de ignorar) los paquetes ya recibidos con determinados números de secuencia *inferiores* al número base actual de la ventana. Puede comprobar fácilmente que este doble reconocimiento es necesario. Por ejemplo, dados los espacios de números de secuencia del emisor y del receptor de la Figura 3.23, si no hay ningún paquete ACK para el paquete `base_emision` propagándose desde el receptor al emisor, finalmente el emisor retransmitirá el paquete `base_emision`, incluso aunque esté claro (¡para nosotros, pero no para el

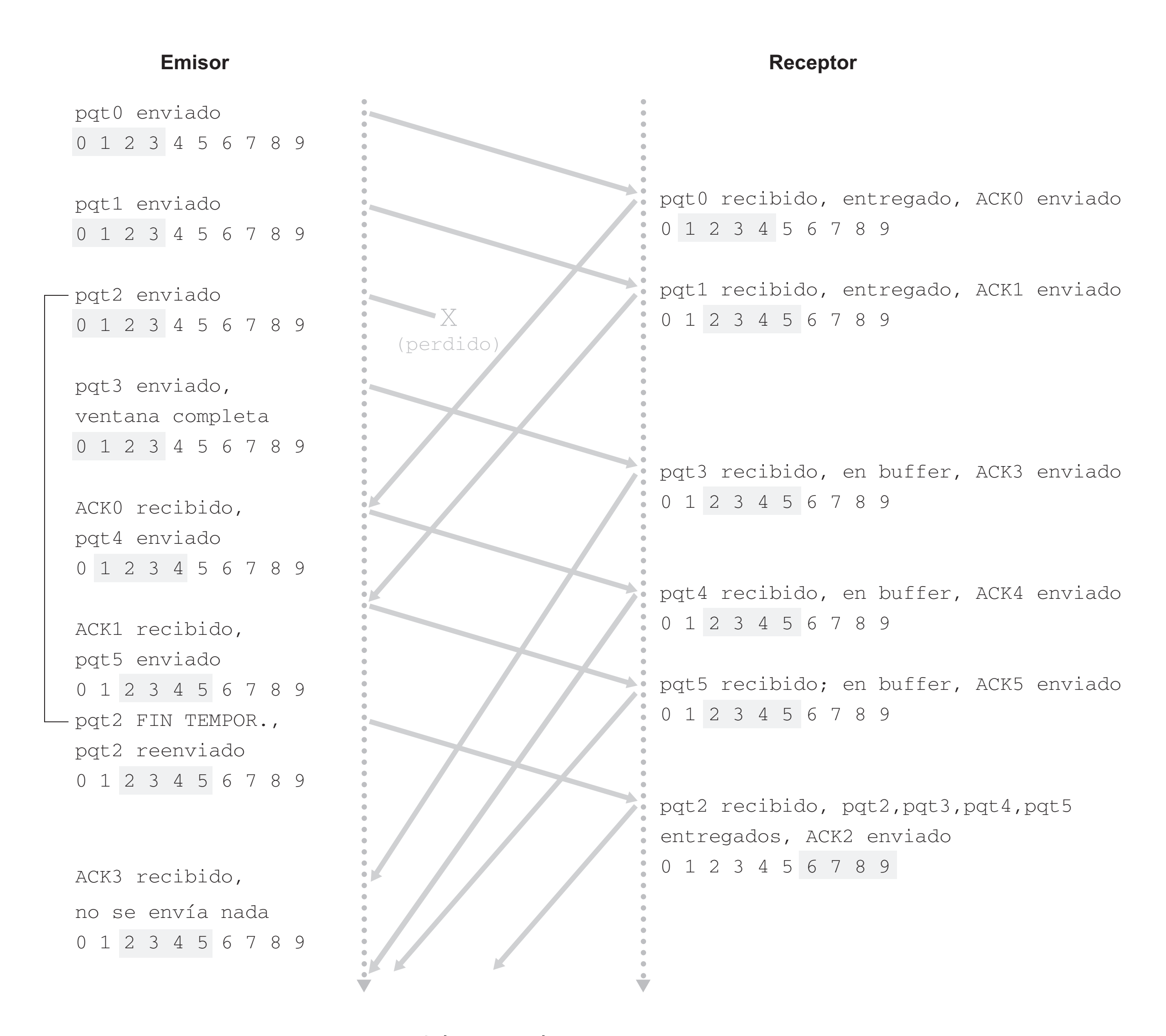
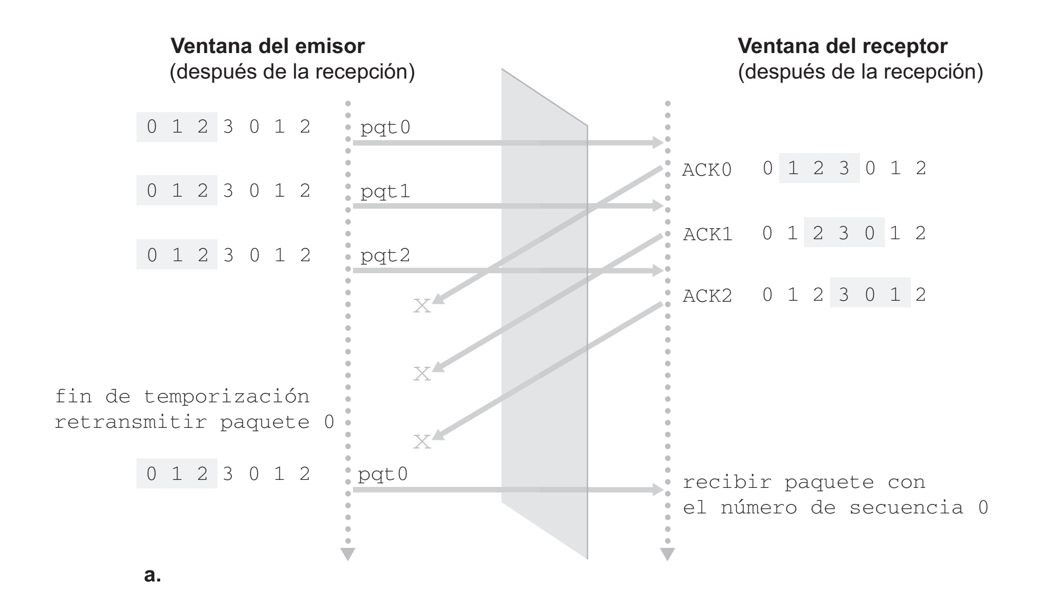


Figura 3.26 • Funcionamiento del protocolo SR.

emisor!) que el receptor ya ha recibido dicho paquete. Si el receptor no hubiera confirmado este paquete, la ventana del emisor nunca avanzaría. Este ejemplo ilustra un aspecto importante de los protocolos SR (y de otros muchos protocolos). El emisor y el receptor no siempre tienen una visión idéntica de lo que se ha recibido correctamente y de lo que no. En los protocolos SR, esto significa que las ventanas del emisor y del receptor no siempre coinciden.

La falta de sincronización entre las ventanas del emisor y del receptor tiene consecuencias importantes cuando nos enfrentamos con la realidad de un rango finito de números de secuencia. Por ejemplo, imagine lo que ocurriría con un rango de cuatro números de secuencia de paquete, 0, 1, 2, 3, y un tamaño de ventana de tres. Suponga que los paquetes 0 a 2 se transmiten y son recibidos correctamente y reconocidos por el receptor. En esta situación, la



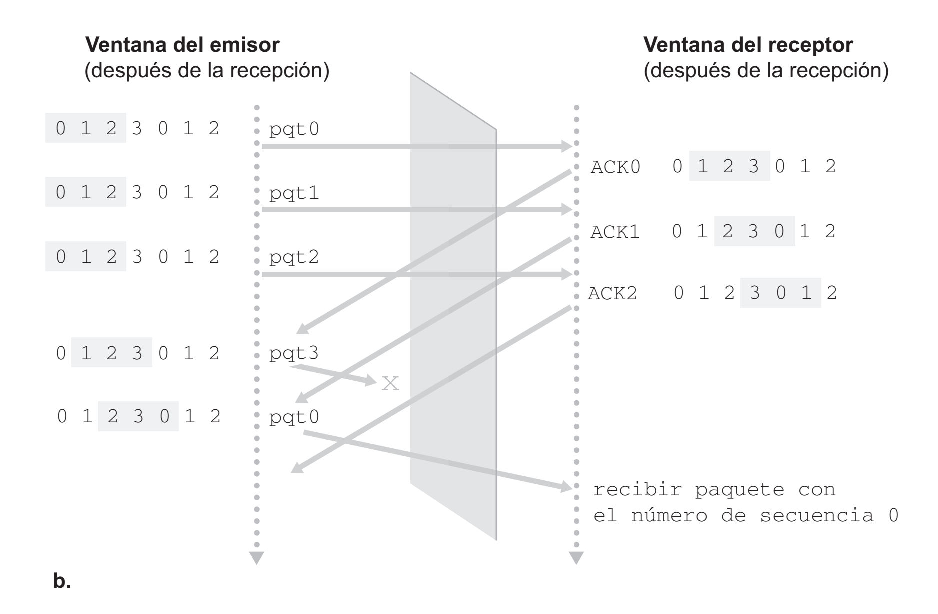


Figura 3.27 • Dilema del receptor de los protocolos SR con ventanas demasiado grandes: ¿un nuevo paquete o una retransmisión?

ventana del receptor se encontraría sobre los paquetes cuarto, quinto y sexto, que tienen los números de secuencia 3, 0 y 1, respectivamente. Ahora consideremos dos escenarios. En el primero, mostrado en la Figura 3.27(a), los mensajes ACK para los tres primeros paquetes se pierden y el emisor los retransmite. A continuación, el receptor recibe un paquete con el número de secuencia 0, una copia del primer paquete enviado.

En el segundo escenario, mostrado en la Figura 3.27(b), los mensajes ACK correspondientes a los tres primeros paquetes se entregan correctamente. El emisor hace avanzar su ventana y envía los paquetes cuarto, quinto y sexto, con los número de secuencia 3, 0 y 1, respectivamente. El paquete con el número de secuencia 3 se pierde pero llega el paquete con el número de secuencia 0, un paquete que contiene datos *nuevos*.

Examinemos ahora el punto de vista del receptor en la Figura 3.27, el cual tiene delante una cortina imaginaria entre el emisor y el receptor, ya que el receptor no puede "ver" las acciones que lleva a cabo el emisor. Todo lo que ve el receptor es la secuencia de mensajes que recibe del canal y que él envía al mismo. En lo que respecta al receptor, los dos escenarios de la Figura 3.27 son *idénticos*. No hay forma de diferenciar la retransmisión del primer paquete de la transmisión inicial del quinto paquete. Evidentemente, un tamaño de ventana que sea una unidad menor que el tamaño del espacio de números de secuencia no puede funcionar. Entonces, ¿cuál tiene que ser el tamaño de la ventana? Uno de los problemas incluidos al final del capítulo le pedirá que demuestre que el tamaño de la ventana tiene que ser menor o igual que la mitad del tamaño del espacio de números de secuencia en los protocolos SR.

En el sitio web de acompañamiento encontrará un applet que simula el funcionamiento del protocolo SR. Intente llevar a cabo los mismos experimentos que realizó con el applet de GBN. ¿Coinciden los resultados con los que cabría esperar?

Con esto hemos terminado con nuestra exposición sobre los protocolos fiables de transferencia de datos. Hemos visto *muchos* de los numerosos mecanismos básicos que contribuyen a proporcionar una transferencia de datos fiable. La Tabla 3.1 resume estos mecanismos. Ahora que ya hemos visto todos estos mecanismos en funcionamiento y que hemos adquirido una "visión de conjunto", le animanos a que repase esta sección con el fin de ver cómo estos mecanismos se fueron añadiendo para cubrir los modelos cada vez más complejos (y realistas) del canal que conecta al emisor y el receptor, o para mejorar el rendimiento de los protocolos.

Concluimos esta exposición sobre los protocolos de transferencia de datos fiables haciendo una suposición más sobre el modelo del canal subyacente. Recuerde que hemos supuesto que los paquetes no se pueden reordenar dentro del canal existente entre el emisor y el receptor. Generalmente, esta suposición es razonable cuando el emisor y el receptor están conectados simplemente mediante un cable físico. Sin embargo, cuando el "canal" que los conecta es una red puede tener lugar la reordenación de paquetes. Una manifestación de la reordenación de paquetes es que pueden aparecer copias antiguas de un paquete con un número de secuencia o de reconocimiento x, incluso aunque ni la ventana del emisor ni la del receptor contengan x. Con la reordenación de paquetes, puede pensarse en el canal como en un buffer que almacena paquetes y que espontáneamente puede transmitirlos en *cualquier* instante futuro. Puesto que los números de secuencia pueden reutilizarse, hay que tener cuidado para prevenir la aparición de esos paquetes duplicados. En la práctica, lo que se hace es asegurarse de que no se reutilice un número de secuencia hasta que el emisor esté "seguro" de que los paquetes enviados anteriormente con el número de secuencia x ya no se encuentran en la red. Esto se hace suponiendo que un paquete no puede "vivir" en la red durante más tiempo que un cierto periodo temporal máximo fijo. En las ampliaciones de TCP para redes de alta velocidad se supone un tiempo de vida máximo de paquete de aproximadamente tres minutos [RFC 1323]. [Sunshine 1978] describe un método para utilizar números de secuencia tales que los problemas de reordenación pueden ser eliminados por completo.

Mecanismo	Uso, comentarios
Suma de comprobación (checksum)	Utilizada para detectar errores de bit en un paquete transmitido.
Temporizador	Se emplea para detectar el fin de temporización y retransmitir un paquete, posiblemente porque el paquete (o su mensaje ACK correspondiente) se ha perdido en el canal. Puesto que se puede producir un fin de temporización si un paquete está retardado pero no perdido (fin de temporización prematura), o si el receptor ha recibido un paquete pero se ha perdido el correspondiente ACK del receptor al emisor, puede ocurrir que el receptor reciba copias duplicadas de un paquete.
Número de secuencia	Se emplea para numerar secuencialmente los paquetes de datos que fluyen del emisor hacia el receptor. Los saltos en los números de secuencia de los paquetes recibidos permiten al receptor detectar que se ha perdido un paquete. Los paquetes con números de secuencia duplicados permiten al receptor detectar copias duplicadas de un paquete.
Reconocimiento (ACK)	El receptor utiliza estos paquetes para indicar al emisor que un paquete o un conjunto de paquetes ha sido recibido correctamente. Los mensajes de reconocimiento suelen contener el número de secuencia del paquete o los paquetes que están confirmando. Dependiendo del protocolo, los mensajes de reconocimiento pueden ser individuales o acumulativos.
Reconocimiento negativo (NAK)	El receptor utiliza estos paquetes para indicar al emisor que un paquete no ha sido recibido correctamente. Normalmente, los mensajes de reconocimiento negativo contienen el número de secuencia de dicho paquete erróneo.
Ventana, procesamiento en cadena	El emisor puede estar restringido para enviar únicamente paquetes cuyo número de secuencia caiga dentro de un rango determinado. Permitiendo que se transmitan varios paquetes aunque no estén todavía reconocidos, se puede incrementar la tasa de utilización del emisor respecto al modo de operación de los protocolos de parada y espera. Veremos brevemente que el tamaño de la ventana se puede establecer basándose en la capacidad del receptor para recibir y almacenar en buffer los mensajes, o en el nivel de congestión de la red, o en ambos parámetros.

Tabla 3.1 • Resumen de los mecanismos para la transferencia de datos fiable y su uso.

3.5 Transporte orientado a la conexión: TCP

Ahora que ya hemos visto los principios básicos de la transferencia de datos fiable, vamos a centrarnos en TCP, un protocolo de la capa de transporte de Internet, fiable y orientado a la conexión. En esta sección veremos que para proporcionar una transferencia de datos fiable, TCP confía en muchos de los principios básicos expuestos en la sección anterior, incluyendo los mecanismos de detección de errores, las retransmisiones, los reconocimientos acumulativos, los temporizadores y los campos de cabecera para los números de secuencia y de reconocimiento. El protocolo TCP está definido en los documentos RFC 793, RFC 1122, RFC 1323, RFC 2018 y RFC 2581.

3.5.1 La conexión TCP

Se dice que TCP **está orientado a la conexión** porque antes de que un proceso de la capa aplicación pueda comenzar a enviar datos a otro, los dos procesos deben primero "establecer una comunicación" entre ellos; es decir, tienen que enviarse ciertos segmentos preliminares para definir los parámetros de la transferencia de datos que van a llevar a cabo a continuación. Como parte del proceso de establecimiento de la conexión TCP, ambos lados de la misma iniciarán muchas variables de estado TCP (muchas de las cuales se verán en esta sección y en la Sección 3.7) asociadas con la conexión TCP.

La "conexión" TCP no es un circuito terminal a terminal con multiplexación TDM o FDM como lo es una red de conmutación de circuitos. Ni tampoco es un circuito virtual (véase el Capítulo 1), ya que el estado de la conexión reside completamente en los dos sistemas terminales. Dado que el protocolo TCP se ejecuta únicamente en los sistemas terminales y no en los elementos intermedios de la red (routers y switches de la capa de enlace), los elementos intermedios de la red no mantienen el estado de la conexión TCP. De hecho, los routers intermedios son completamente inconscientes de las conexiones TCP; los routers ven los datagramas, no las conexiones.

Una conexión TCP proporciona un **servicio full-duplex**: si existe una conexión TCP entre el proceso A que se ejecuta en un host y el proceso B que se ejecuta en otro host, entonces los datos de la capa de aplicación pueden fluir desde el proceso A al proceso B en el mismo instante que los datos de la capa de aplicación fluyen del proceso B al proceso A.



HISTORIA

VINTON CERF, ROBERT KAHN Y TCP/IP

A principios de la década de 1970 comenzaron a proliferar las redes de conmutación de paquetes, siendo ARPAnet (la red precursora de Internet) sólo una más de muchas redes. Cada una de estas redes utilizaba su propio protocolo. Dos investigadores, Vinton Cerf y Robert Kahn, se dieron cuenta de la importancia de interconectar estas redes e inventaron un protocolo inter-red denominado TCP/IP (*Transmission Control Protocol/Internet Protocol*). Aunque Cerf y Kahn comenzaron viendo el protocolo como una sola entidad, más tarde la dividieron en dos partes, TCP e IP, que operaban por separado. Cerf y Kahn publicaron un informe sobre TCP/IP en mayo de 1974 en *IEEE Transactions on Communications Technology* [Cerf 1974].

El protocolo TCP/IP, que es la base de la Internet actual, fue diseñado antes que los PC y las estaciones de trabajo, antes de la proliferación de las tecnologías de las redes Ethernet y otras redes de área local, y antes que la Web, los flujos de audio y los chat. Cerf y Kahn vieron la necesidad que existía de un protocolo de red que, por un lado, proporcionara un amplio soporte para las aplicaciones que ya estaban definidas y que, por otro lado, permitiera interoperar a los hosts y los protocolos de la capa de enlace.

En 2004, Cerf y Kahn recibieron el premio Turing Award de ACM, que está considerado como el "Premio Nobel de la Informática" por su trabajo pionero sobre los procesos de comunicación entre redes, incluyendo el diseño y la implementación de los protocolos básicos de comunicación de Internet (TCP/IP) y por su liderazgo en el mundo de las redes.

Una conexión TCP casi siempre es una conexión **punto a punto**, es decir, entre un único emisor y un único receptor. La "multidifusión" (véase la Sección 4.7), la transferencia de datos desde un emisor a muchos receptores en una única operación, no es posible con TCP. Con TCP, dos hosts son compañía y tres multitud.

Veamos ahora cómo se establece una conexión TCP. Suponga que un proceso que se está ejecutando en un host desea iniciar una conexión con otro proceso que se ejecuta en otro host. Recuerde que el proceso que inicia la conexión es el *proceso cliente*, y el otro proceso es el *proceso servidor*. El proceso de la aplicación cliente informa en primer lugar a la capa de transporte del cliente que desea establecer una conexión con un proceso del servidor. Recuerde que en la Sección 2.7, hemos visto un programa cliente en Java que hacía esto ejecutando el comando:

```
Socket socketCliente = new Socket("nombrehost", NúmeroPuerto);
```

donde `nombrehost` es el nombre del servidor y `NumeroPuerto` identifica al proceso del servidor. La capa de transporte del cliente procede entonces a establecer una conexión TCP con el TCP del servidor. Al finalizar esta sección veremos en detalle el procedimiento de establecimiento de la conexión. Por el momento, nos basta con saber que el cliente primero envía un segmento TCP especial; el servidor responde con un segundo segmento TCP especial y, por último, el cliente responde de nuevo con tercer segmento especial. Los dos primeros segmentos no transportan ninguna carga útil; es decir, no transportan datos de la capa de aplicación; el tercero de estos segmentos es el que puede llevar la carga útil. Puesto que los tres segmentos son intercambiados entre dos hosts, este procedimiento de establecimiento de la conexión suele denominarse **acuerdo en tres fases**.

Una vez que se ha establecido una conexión TCP, los dos procesos de aplicación pueden enviarse datos el uno al otro. Consideremos la transmisión de datos desde el proceso cliente al proceso servidor. El proceso cliente pasa un flujo de datos a través del socket (la puerta del proceso), como se ha descrito en la Sección 2.7. Una vez que los datos atraviesan la puerta, se encuentran en manos del protocolo TCP que se ejecuta en el cliente. Como se muestra en la Figura 3.28, TCP dirige estos datos al **buffer de emisión** de la conexión, que es uno de los buffers que se definen durante el proceso inicial del acuerdo en tres fases. De vez en cuando, TCP tomará fragmentos de datos del buffer de emisión. La especificación de TCP [RFC 793] es bastante vaga en lo que respecta a especificar cuándo TCP debe realmente enviar los datos almacenados en el buffer, enunciando que TCP "debe transmitir esos datos en segmentos según su propia conveniencia". La cantidad máxima de datos que pueden cogerse y colocarse en un segmento está limitada por el **tamaño máximo de segmento (MSS, *Maximum Segment Size*)**. Normalmente, el MSS queda determinado en primer lugar por la longitud de la trama más larga de la capa de enlace que el host emisor local puede enviar [que es la **unidad máxima de transmisión, (MTU, *Maximum Transmission Unit*)**], y luego el MSS se establece de manera que se garantice que un segmento TCP (cuando se encapsula en un datagrama IP) se ajuste a una única trama de la capa de enlace. Valores comunes de MTU son 1.460 bytes, 536 bytes y 512 bytes. También se han propuesto métodos para descubrir la MTU de la ruta (la trama más larga de la capa de enlace que puede enviarse a través de todos los enlaces desde el origen hasta el destino) [RFC 1191] y establecer el MSS basándose en el valor de la MTU de la ruta. Observe que el MSS es la cantidad máxima de datos de la capa de aplicación en el segmento, no el tamaño máximo del segmento TCP incluyendo las cabeceras. Esta terminología resulta confusa, pero tenemos que convivir con ella, porque está muy extendida.

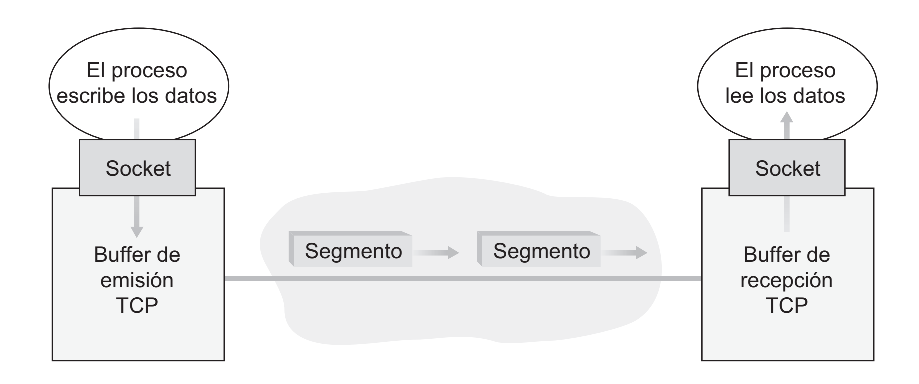


Figura 3.28 • Buffers de emisión y recepción de TCP.

TCP empareja cada fragmento de datos del cliente con una cabecera TCP, formando **segmentos TCP**. Los segmentos se pasan a la capa de red, donde son encapsulados por separado dentro de datagramas IP de la capa de red. Los datagramas IP se envían entonces a la red. Cuando TCP recibe un segmento en el otro extremo, los datos del mismo se colocan en el buffer de recepción de la conexión TCP, como se muestra en la Figura Figure 3.28. La aplicación lee el flujo de datos de este buffer. Cada lado de la conexión tiene su propio buffer de emisión y su propio buffer de recepción (puede ver el applet de control de flujo en línea en http://www.awl.com/kurose-ross, que proporciona una animación de los buffers de emisión y de recepción).

Por tanto, una conexión TCP consta de buffers, variables y un socket de conexión a un proceso en un host, y otro conjunto de buffers, variables y un socket de conexión a un proceso en otro host. Como hemos mencionado anteriormente, no se asignan ni buffers ni variables a la conexión dentro de los elementos de red (routers, switches y repetidores) existentes entre los hosts.

3.5.2 Estructura del segmento TCP

Después de haber visto de forma breve la conexión TCP, examinemos la estructura de un segmento TCP. El segmento TCP consta de campos de cabecera y un campo de datos. El campo de datos contiene un fragmento de los datos de la aplicación. Como hemos mencionado anteriormente, el MSS limita el tamaño máximo del campo de datos de un segmento. Cuando TCP envía un archivo grande, como por ejemplo una imagen como parte de una página web, normalmente divide el archivo en fragmentos de tamaño MSS (excepto el último fragmento, que normalmente será más pequeño que MSS). Sin embargo, las aplicaciones interactivas suelen transmitir fragmentos de datos que son más pequeños que el MSS; por ejemplo, en las aplicaciones de inicio de sesión remoto (*remote login*) como Telnet, el campo de datos del segmento TCP sólo tiene un byte. Puesto que habitualmente la cabecera de TCP tiene 20 bytes (12 bytes más que la cabecera de UDP), los segmentos enviados mediante Telnet sólo pueden tener una longitud de 21 bytes.

La Figura 3.29 muestra la estructura del segmento TCP. Al igual que con UDP, la cabecera incluye los **número de puerto de origen y de destino**, que se uti[redacted]an para multiplexar y demultiplexar los datos de y para las aplicaciones de la capa superior. También, al igual que UDP, la cabecera incluye un **campo de suma de comprobación**. La cabecera de un segmento TCP también contiene los siguientes campos:

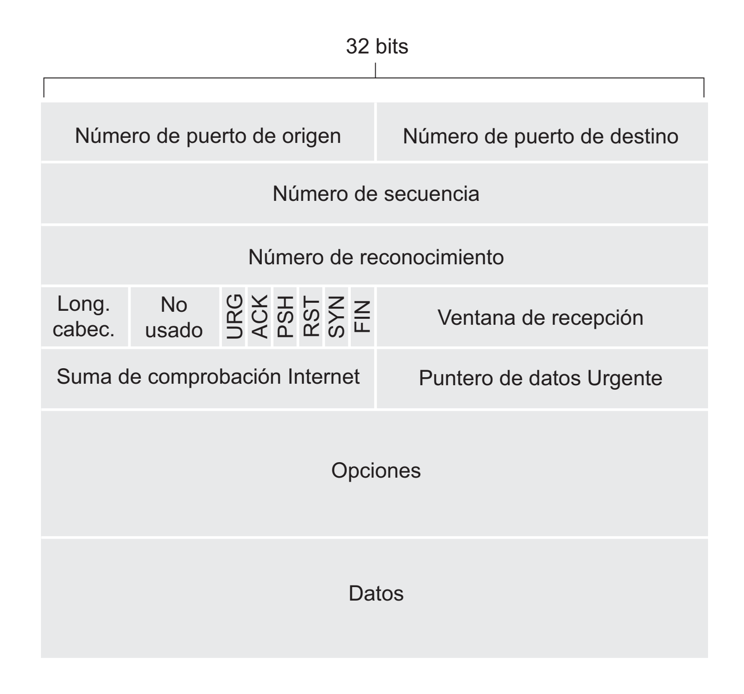


Figura 3.29 • Estructura del segmento TCP.

- El **campo número de secuencia** de 32 bits y el **campo número de reconocimiento** también de 32 bits son utilizados por el emisor y el receptor de TCP para implementar un servicio de transferencia de datos fiable, como se explica más adelante.
- El campo **ventana de recepción** de 16 bits se utiliza para el control de flujo. Veremos en breve que se emplea para indicar el número de bytes que un receptor está dispuesto a aceptar.
- El **campo longitud de cabecera** de 4 bits especifica la longitud de la cabecera TCP en palabras de 32 bits. La cabecera TCP puede tener una longitud variable a causa del campo opciones de TCP (normalmente, este campo está vacío, por lo que la longitud de una cabecera TCP típica es de 20 bytes).
- El **campo opciones** es opcional y de longitud variable. Se utiliza cuando un emisor y un receptor negocian el tamaño máximo de segmento (MSS) o como un factor de escala de la ventana en las redes de alta velocidad. También se define una opción de marca temporal. Consulte los documentos RFC 854 y RFC 1323 para conocer detalles adicionales.
- El **campo indicador** tiene 6 bits. El **bit ACK** se utiliza para indicar que el valor transportado en el campo de reconocimiento es válido; es decir, el segmento contiene un reconocimiento para un segmento que ha sido recibido correctamente. Los bits **RST**, **SYN** y **FIN** se utilizan para el establecimiento y cierre de conexiones, como veremos al final de esta sección. La activación del bit **PSH** indica que el receptor deberá pasar los datos a la capa superior de forma inmediata. Por último, el bit **URG** se utiliza para indicar que hay datos en este segmento que la entidad de la capa superior del lado emisor ha marcado como "urgentes". La posición de este último byte de estos datos urgentes se indica mediante el **campo puntero de datos urgentes** de 16 bits. TCP tiene que informar a la

entidad de la capa superior del lado receptor si existen datos urgentes y pasarle un puntero a la posición donde finalizan los datos urgentes. En la práctica, PSH, URG y el puntero a datos urgentes no se utilizan. Sin embargo, hemos hablado de estos campos con el fin de proporcionar al lector la información completa.

Números de secuencia y números de reconocimiento

Dos de los campos más importantes de la cabecera de un segmento TCP son el campo número de secuencia y el campo número de reconocimiento. Estos campos son una parte crítica del servicio de transferencia de datos fiable de TCP. Pero antes de ver cómo se utilizan estos campos para proporcionar una transferencia de datos fiable, explicaremos en primer lugar lo que pone exactamente TCP en esos campos.

TCP percibe los datos como un flujo de bytes no estructurado pero ordenado. El uso que hace TCP de los números de secuencia refleja este punto de vista, en el sentido de que los números de secuencia hacen referencia al flujo de bytes transmitido y *no* a la serie de segmentos transmitidos. El **número de secuencia de un segmento** es por tanto el número del primer byte del segmento dentro del flujo de bytes. Veamos un ejemplo. Suponga que un proceso del host A desea enviar un flujo de datos a un proceso del host B a través de una conexión TCP. El protocolo TCP en el host A numerará implícitamente cada byte del flujo de datos. Suponga también que el flujo de datos consta de un archivo de 500.000 bytes, que el tamaño MSS es de 1.000 bytes y que el primer byte del flujo de datos está numerado como 0. Como se muestra en la Figura 3.30, TCP construye 500 segmentos a partir del flujo de datos. El primer segmento tiene asignado el número de secuencia 0, el segundo segmento tiene asignado el número de secuencia 1.000, el tercero tendrá asignado el número de secuencia 2.000, etc. Cada número de secuencia se inserta en el campo número de secuencia de la cabecera del segmento TCP apropiado.

Consideremos ahora los números de reconocimiento, que son algo más complicados que los números de secuencia. Recuerde que TCP es una conexión full-duplex, de modo que el host A puede estar recibiendo datos del host B mientras envía datos al host B (como parte de la misma conexión TCP). Todos los segmentos que llegan procedentes del host B tienen un número de secuencia para los datos que fluyen de B a A. *El número de reconocimiento que el host A incluye en su segmento es el número de secuencia del siguiente byte que el host A está esperando del host B*. Veamos algunos ejemplos para comprender qué es lo que ocurre aquí. Suponga que el host A ha recibido todos los bytes numerados de 0 a 535 procedentes de B y suponga también que está enviando un segmento al host B. El host A está esperando al 536 y todos los bytes que le siguen del flujo de datos del host B. Por tanto, el host A incluye 536 en el campo número de reconocimiento del segmento que envía a B.

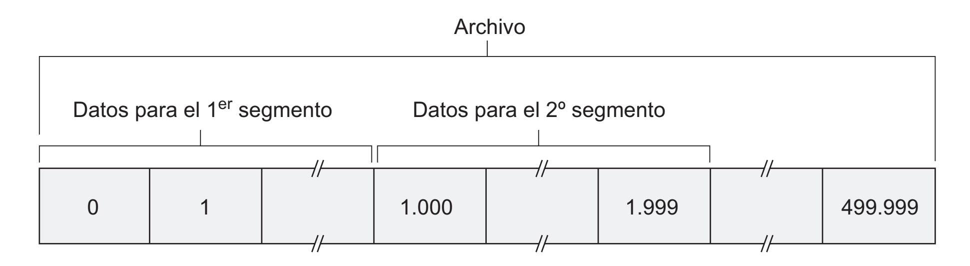


Figura 3.30 • División de los datos del archivo en segmentos TCP.

Otro ejemplo: suponga que el host A ha recibido un segmento del host B que contiene los bytes de 0 a 535 y otro segmento que contiene los bytes de 900 a 1.000. Por alguna razón, el host A no ha recibido todavía los bytes de 536 a 899. En este ejemplo, el host A está esperando el byte 536 (y los que le siguen) para volver a crear el flujo de datos de B. Por tanto, el siguiente segmento de A a B contendrá el número 536 en el campo de número de reconocimiento. Dado que TCP sólo confirma los bytes hasta el primer byte que falta en el flujo, se dice que TCP proporciona **reconocimientos acumulativos**.

Este último ejemplo plantea también un problema importante aunque sutil. El host A ha recibido el tercer segmento (bytes 900 a 1.000) antes de recibir el segundo segmento (bytes 536 a 899). Por tanto, el tercer segmento no ha llegado en orden. El sutil problema es el siguiente: ¿qué hace un host cuando no recibe los segmentos en orden a través de una conexión TCP? Curiosamente, los RFC dedicados a TCP no imponen ninguna regla y dejan la decisión a las personas que programan las implementaciones de TCP. Básicamente, tenemos dos opciones: (1) el receptor descarta de forma inmediata los segmentos que no han llegado en orden (lo que, como hemos anteriormente, puede simplificar el diseño del receptor) o (2) el receptor mantiene los bytes no ordenados y espera a que lleguen los bytes que faltan con el fin de rellenar los huecos. Evidentemente, esta última opción es más eficiente en términos de ancho de banda de la red, y es el método que se utiliza en la práctica.

En la Figura 3.30 hemos supuesto que el número de secuencia inicial era cero. En la práctica, ambos lados de una conexión TCP eligen aleatoriamente un número de secuencia inicial. Esto se hace con el fin de minimizar la posibilidad de que un segmento que todavía está presente en la red a causa de una conexión anterior que ya ha terminado entre dos hosts pueda ser confundido con un segmento válido de una conexión posterior entre esos dos mismos hosts (que también estén usando los mismos números de puerto que la conexión antigua) [Sunshine 1978].

Telnet: caso de estudio de los números de secuencia y de reconocimiento

Telnet, definido en el documento RFC 854, es un popular protocolo de la capa de aplicación utilizado para los inicios de sesión remotos. Se ejecuta sobre TCP y está diseñado para trabajar entre cualquier pareja de hosts. A diferencia de las aplicaciones de transferencia masiva de datos vistas en el Capítulo 2, Telnet es una aplicación interactiva. Vamos a ver aquí un ejemplo, ya que ilustra muy bien los números de secuencia y de reconocimiento de TCP. Debemos mencionar que actualmente muchos usuarios prefieren utilizar el protocolo SSH en lugar de Telnet, porque los datos enviados a través de una conexión Telnet (¡incluidas las contraseñas!) no están cifrados, lo que hace que Telnet sea vulnerable a los ataques de personas que quieran escuchar la conexión (como veremos en la Sección 8.7).

Supongamos que el host A inicia una sesión Telnet con el host B. Puesto que el host A inicia la sesión, se etiqueta como el cliente y el host B como el servidor. Cada carácter escrito por el usuario (en el cliente) se enviará al host remoto; el host remoto devolverá una copia de cada carácter, que será mostrada en la pantalla del usuario Telnet. Este "eco" se emplea para garantizar que los caracteres vistos por el usuario Telnet ya han sido recibidos y procesados en el sitio remoto. Por tanto, cada carácter atraviesa la red dos veces entre el instante en el que usuario pulsa una tecla y el instante en el que el carácter se muestra en el monitor del usuario.

Suponga ahora que el usuario escribe una única letra, la 'C', y luego se va a por un café. Examinemos los segmentos TCP que están siendo enviados entre el cliente y el servidor. Como se muestra en la Figura 3.31, suponemos que los números de secuencia iniciales para

el cliente y el servidor son, respectivamente, 42 y 79. Recuerde que el número de secuencia de un segmento es el número de secuencia del primer byte del campo de datos. Por tanto, el primer segmento enviado por el cliente tendrá el número de secuencia 42 y el primer segmento enviado desde el servidor tendrá el número de secuencia 79. Recuerde que el número de reconocimiento es el número de secuencia del siguiente byte de datos que el host está esperando. Cuando ya se ha establecido una conexión TCP pero todavía no se enviado ningún dato, el cliente está esperando la llegada del byte 79 y el servidor está esperando al byte 42.

Como se muestra en la Figura 3.31, se envían tres segmentos. El primer segmento se transmite desde el cliente al servidor, conteniendo la representación ASCII de 1 byte de la letra 'C' en su campo de datos. Este primer segmento también contiene el número 42 en su campo número de secuencia, como ya hemos descrito. Además, dado que el cliente todavía no ha recibido ningún dato procedente del servidor, este primer segmento contendrá el número 79 en su campo número de reconocimiento.

El segundo segmento se envía desde el servidor al cliente y además sirve a un doble propósito. En primer lugar, proporciona un reconocimiento de los datos que ha recibido el servidor. Al incluir el número 43 en el campo número de reconocimiento, el servidor está diciendo al cliente que ha recibido correctamente todo hasta el byte 42 y ahora está esperando los bytes 43 y posteriores. El segundo propósito de este segmento es devolver el eco de la letra 'C'. Por tanto, el segundo segmento contiene la representación ASCII de la letra 'C' en su campo de datos. Este segundo segmento tiene el número de secuencia 79, el número de secuencia inicial del flujo de datos servidor-cliente de esta conexión TCP, ya que

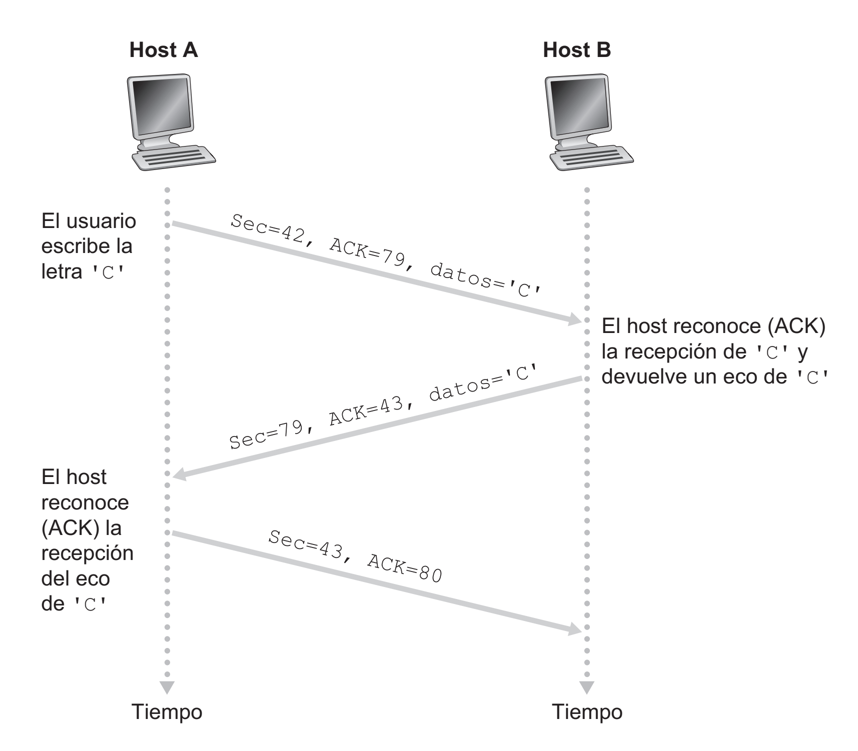


Figura 3.31 • Números de secuencia y de reconocimiento en una aplicación Telnet simple sobre TCP.

se trata del primer byte de datos que el servidor está enviando. Observe que la confirmación de los datos enviados desde el cliente al servidor se transporta en un segmento que contiene los datos enviados del servidor al cliente; se dice que este reconocimiento está **superpuesto** al segmento de datos enviado por el servidor al cliente.

El tercer segmento se envía desde el cliente al servidor. Su único propósito es confirmar los datos que ha recibido procedentes del servidor (recuerde que el segundo segmento contenía datos, la letra 'C', del servidor para el cliente). Este segmento tiene un campo de datos vacío (es decir, el paquete de reconocimiento no va superpuesto a los datos que van del cliente al servidor). El segmento contiene el número 80 en el campo número de reconocimiento porque el cliente ha recibido el flujo de bytes hasta el byte con el número de secuencia 79 y ahora está esperando los bytes 80 y subsiguientes. Es posible que le parezca extraño que este segmento también tenga un número de secuencia aunque no contenga datos pero, dado que los segmentos TCP disponen de un campo número de secuencia, es necesario incluir siempre en ese campo un valor.

3.5.3 Estimación del tiempo de ida y vuelta y fin de temporización

TCP, al igual que nuestro protocolo `rdt` de la Sección 3.4, utiliza un mecanismo de fin de temporización/retransmisión para recuperarse de la pérdida de segmentos. Aunque conceptualmente esto es muy simple, surgen muchos problemas sutiles al implementar dicho mecanismo en un protocolo real como, por ejemplo, TCP. Quizá la cuestión más obvia es la longitud de los intervalos de fin de temporización. Evidentemente, el intervalo de fin de temporización debería ser mayor que el tiempo de ida y vuelta (RTT) de la conexión; es decir, mayor que el tiempo que transcurre desde que se envía un segmento hasta que se recibe su reconocimiento. Si fuera de otra manera, se enviarían retransmisiones innecesarias. Pero, ¿cuánto mayor? y, ¿cómo se debería estimar el RTT por primera vez? ¿Debería asociarse un temporizador con cada uno de los segmentos no reconocidos? ¡Demasiadas preguntas! En esta sección vamos a basar nuestra exposición en el trabajo sobre TCP de [Jacobson 1988] y en las recomendaciones actuales del IETF para gestionar los temporizadores TCP [RFC 2988].

Estimación del tiempo de ida y vuelta

Comencemos nuestro estudio de la gestión del temporizador TCP viendo cómo estima TCP el tiempo de ida y vuelta entre el emisor y el receptor. Esto se lleva a cabo de la siguiente manera: el RTT de muestra, expresado como `RTTMuestra`, para un segmento es la cantidad de tiempo que transcurre desde que se envía el segmento (es decir, se pasa a IP) hasta que se recibe el correspondiente paquete de reconocimiento del segmento. En lugar de medir `RTTMuestra` para cada segmento transmitido, la mayor parte de las implementaciones TCP toman sólo una medida de `RTTMuestra` cada vez. Es decir, en cualquier instante, `RTTMuestra` se estima a partir de uno solo de los segmentos transmitidos pero todavía no reconocidos, lo que nos proporciona un nuevo valor de `RTTMuestra` aproximadamente cada RTT segundos. Además, TCP nunca calcula `RTTMuestra` para un segmento que haya sido retransmitido; sólo mide este valor para los segmentos que han sido transmitidos una vez. En uno de los problemas incluidos al final del capítulo se le pedirá que explique el por qué de este comportamiento.

Obviamente, los valores de `RTTMuestra` fluctuarán de un segmento a otro a causa de la congestión en los routers y a la variación de la carga en los sistemas terminales. A causa de esta fluctuación, cualquier valor de `RTTMuestra` dado puede ser atípico. Con el fin de estimar un RTT típico, es natural por tanto calcular algún tipo de promedio de los valores de `RTTMuestra`. TCP mantiene un valor promedio, denominado `RTTEstimado`, de los valores `RTTMuestra`. Para obtener un nuevo valor de `RTTMuestra`, TCP actualiza `RTTEstimado` según la fórmula siguiente:

```
RTTEstimado = (1 – α) · RTTEstimado + α · RTTMuestra
```

Hemos escrito esta fórmula como una instrucción de un lenguaje de programación (el nuevo valor de `RTTEstimado` es una combinación ponderada del valor anterior de `RTTEstimado` y del nuevo valor de `RTTMuestra`). El valor recomendado para α es 0,125 (es decir, 1/8) [RFC 2988], en cuyo caso la fórmula anterior se expresaría como sigue:

```
RTTEstimado = 0,875 · RTTEstimado + 0,125 · RTTMuestra
```

Observe que `RTTEstimado` es una media ponderada de los valores de `RTTMuestra`. Como se examina en uno de los problemas de repaso incluidos al final del capítulo, esta media ponderada asigna un mayor peso a las muestras recientes que a las más antiguas. Esto es lógico, ya que las muestras más recientes reflejan mejor la congestión que existe actual-



PRÁCTICA

TCP proporciona un servicio de transferencia de datos fiable utilizando mensajes de reconocimiento positivos y temporizadores, de forma muy similar a como hemos visto en la Sección 3.4. TCP confirma los datos que ha recibido correctamente y retransmite segmentos cuando éstos o sus correspondientes reconocimientos se piensa que se han perdido o se han corrompido. Ciertas versiones de TCP también disponen de un mecanismo NAK implícito con un mecanismo rápido de retransmisión TCP: la recepción de tres ACK duplicados para un determinado segmento sirve como un NAK implícito para el siguiente segmento, provocando la retransmisión de dicho segmento antes del fin de la temporización. TCP utiliza secuencias de números para permitir al receptor identificar los segmentos perdidos o duplicados. Al igual que en el caso de nuestro protocolo de transferencia de datos fiable, `rdt3.0`, TCP no puede por sí mismo saber si un cierto segmento, o su correspondiente ACK, se ha perdido, está corrompido o se ha retardado demasiado. En el emisor, la respuesta TCP será la misma: retransmitir el segmento en cuestión.

TCP también utiliza el procesamiento en cadena, permitiendo al emisor tener múltiples segmentos transmitidos pero aun no reconocidos en cualquier instante. Anteriormente hemos visto que el procesamiento en cadena puede mejorar enormemente la tasa de transferencia de una sesión cuando la relación entre el tamaño del segmento y el retardo de ida y vuelta es pequeña. El número específico de segmentos pendientes no reconocidos que un emisor puede tener se determina mediante los mecanismos de control de congestión y de control de flujo de TCP. El control de flujo de TCP se examina al final de esta sección y el mecanismo de control de congestión en la Sección 3.7. Por el momento, basta con que seamos conscientes de que el emisor TCP utiliza el procesamiento en cadena.

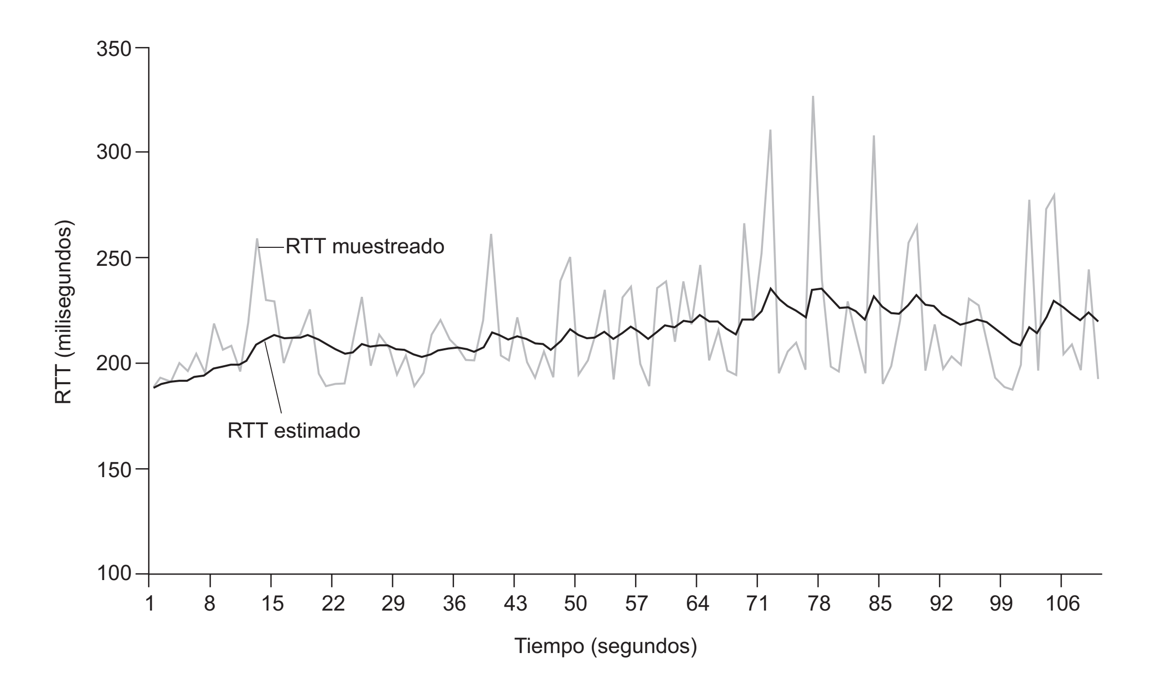


Figura 3.32 • Muestreo de RTT y estimación de RTT.

mente en la red. En estadística, una media como ésta se denomina **Media móvil exponencialmente ponderada** (**EWMA,** ***Exponential Weighted Moving Average***). El término "exponencial" aparece en EWMA porque el peso de un valor dado de `RTTMuestra` disminuye exponencialmente tan rápido como tienen lugar las actualizaciones. En los problemas de repaso se le pedirá que deduzca el término exponencial en `RTTEstimado`.

La Figura 3.32 muestra los valores `RTTMuestra` y `RTTEstimado` para $\alpha = 1/8$ en una conexión TCP entre `gaia.cs.umass.edu` (en Amherst, Massachusetts) y `fantasia.eurecom.fr` (en el sur de Francia). Evidentemente, las variaciones en `RTTMuestra` se suavizan en el cálculo de `RTTEstimado`.

Además de tener un estimado de RTT, también es importante disponer de una medida de la variabilidad de RTT. [RFC 2988] define la variación de RTT, `RTTDesv`, como una estimación de cuánto se desvía típicamente `RTTMuestra` de `RTTEstimado`:

```
RTTDesv = (1 – β) · RTTDesv + β·| RTTMuestra – RTTEstimado |
```

Observe que `RTTDesv` es una media EWMA de la diferencia entre `RTTMuestra` y `RTTEstimado`. Si los valores de `RTTMuestra` presentan una pequeña fluctuación, entonces `RTTDesv` será pequeño; por el contrario, si existe una gran fluctuación, `RTTDesv` será grande. El valor recomendado para β es 0,25.

Definición y gestión del intervalo de fin de temporización para la retransmisión

Dados los valores de `RTTEstimado` y `RTTDesv`, ¿qué valor debería utilizarse para el intervalo de fin de temporización de TCP? Evidentemente, el intervalo tendrá que ser mayor o

igual que `RTTEstimado`, o se producirán retransmisiones innecesarias. Pero el intervalo de fin de temporización no debería ser mucho mayor que `RTTEstimado`; de otra manera, si un segmento se pierde, TCP no retransmitirá rápidamente el segmento, provocando retardos muy largos en la transferencia de datos. Por tanto, es deseable hacer el intervalo de fin de temporización igual a `RTTEstimado` más un cierto margen. El margen deberá ser más grande cuando la fluctuación en los valores de `RTTMuestra` sea grande y más pequeño cuando la fluctuación sea pequeña. El valor de `RTTDesv` deberá entonces tenerse en cuenta. Todas estas consideraciones se tienen en cuenta en el método TCP para determinar el intervalo de fin de temporización de las retransmisiones:

```
IntervaloFindeTemporizacion = RTTEstimado + 4 · RTTDesv
```

3.5.4 Transferencia de datos fiable

Recuerde que el servicio de la capa de red de Internet (el servicio IP) no es fiable. IP no garantiza la entrega de los datagramas, no garantiza la entrega en orden de los datagramas y no garantiza la integridad de los datos contenidos en los datagramas. Con el servicio IP, los datagramas pueden desbordar los buffers de los routers y no llegar nunca a su destino, pueden llegar desordenados y los bits de un datagrama pueden corromperse (bascular de 0 a 1, y viceversa). Puesto que los segmentos de la capa de transporte son transportados a través de la red por los datagramas IP, estos segmentos de la capa de transporte pueden sufrir también estos problemas.

TCP crea un **servicio de transferencia de datos fiable** sobre el servicio de mejor esfuerzo pero no fiable de IP. El servicio de transferencia de datos fiable de TCP garantiza que el flujo de datos que un proceso extrae de su buffer de recepción TCP no está corrompido, no contiene huecos, ni duplicados y está en orden; es decir, el flujo de bytes es exactamente el mismo flujo que fue enviado por el sistema terminal existente en el otro extremo de la conexión. La forma en que TCP proporciona una transferencia de datos fiable implica muchos de los principios que hemos estudiado en la Sección 3.4.

En el anterior desarrollo que hemos realizado sobre las técnicas que proporcionan una transferencia de datos fiable, conceptualmente era fácil suponer que cada segmento transmitido pero aun no reconocido tenía asociado un temporizador individual. Aunque esto está bien en teoría, la gestión del temporizador puede requerir una sobrecarga considerable. Por tanto, los procedimientos de gestión del temporizador TCP recomendados [RFC 2988] utilizan un *único* temporizador de retransmisión, incluso aunque haya varios segmentos transmitidos y aún no reconocidos. El protocolo TCP descrito en esta sección sigue esta recomendación de emplear un único temporizador.

Ahora vamos a ver cómo proporciona TCP el servicio de transferencia de datos fiable en dos pasos incrementales. En primer lugar, vamos a presentar una descripción extremadamente simplificada de un emisor TCP que sólo emplea los fines de temporización para recuperarse de las pérdidas de segmentos; a continuación, veremos una descripción más completa que utiliza los mensajes de reconocimiento duplicados además de los fines de temporización. En la siguiente exposición suponemos que sólo se están enviando datos en una sola dirección, del host A al host B, y que el host A está enviando un archivo grande.

La Figura 3.33 presenta una descripción simplificada de un emisor TCP. Vemos que hay tres sucesos importantes relacionados con la transmisión y la retransmisión de datos en el emisor TCP: los datos recibidos desde la aplicación; el fin de temporización del temporizador y la recepción de paquetes ACK. Al producirse el primero de los sucesos más importan-

```
/* Suponga que el emisor no está restringido por los mecanismos de
control de flujo o de control de congestión de TCP, que el tamaño de
los datos procedentes de la capa superior es menor que el MSS y además
la transferencia de datos tiene lugar en un único sentido. */

SigNumSec = NumeroSecuenciaInicial
BaseEmision = NumeroSecuenciaInicial

loop (siempre) {
    switch(suceso)

        suceso: datos recibidos de la aplicación de la capa superior
            crear segmento TCP con número de secuencia SigNumSec
            if (el temporizador no se está ejecutando actualmente)
                iniciar temporizador
            pasar segmento a IP
            SigNumSec = SigNumSec+longitud(datos)
            break;

        suceso: fin de temporización del temporizador
            retransmitir el segmento aun no reconocido con
                el número de secuencia más pequeño
            iniciar temporizador
            break;

        suceso: ACK recibido, con valor de campo ACK igual a y
            if (y > BaseEmision) {
                BaseEmision = y
                if (existen actualmente segmentos aun no reconocidos)
                    iniciar temporizador
                }
            break;

    } /* fin del bucle siempre */
```

Figura 3.33 • Emisor TCP simplificado.

tes TCP recibe datos de la aplicación, encapsula los datos en un segmento y pasa el segmento a IP. Observe que cada segmento incluye un número de secuencia que es el número del primer byte de datos del segmento, dentro del flujo de datos, como se ha descrito en la Sección 3.5.2. Fíjese también en que si el temporizador no está ya funcionando para algún otro segmento, TCP lo inicia en el momento de pasar el segmento a IP (resulta útil pensar en el temporizador como si estuviera asociado con el segmento no reconocido más antiguo). El intervalo de caducidad para este temporizador es `IntervaloFindeTemporizacion`, que se calcula a partir de `RTTEstimado` y `RTTDesv`, como se ha descrito en la Sección 3.5.3.

El segundo suceso importante es el fin de temporización. TCP responde a este suceso retransmitiendo el segmento que ha causado el fin de la temporización y, a continuación, reinicia el temporizador.

El tercer suceso importante que tiene que gestionar el emisor TCP es la llegada de un segmento de reconocimiento (ACK) procedente del receptor (más específicamente, un seg-

mento que contenga un valor de campo ACK válido). Al ocurrir este suceso, TCP compara el valor ACK `y` con su variable `BaseEmision`. La variable de estado TCP `BaseEmision` es el número de secuencia del byte de reconocimiento más antiguo. (Por tanto, `BaseEmision–1` es el número de secuencia del último byte que se sabe que ha sido recibido correctamente y en orden en el receptor). Como hemos indicado anteriormente, TCP utiliza reconocimientos acumulativos, de modo que `y` confirma la recepción de todos los bytes anteriores al número de byte `y`. Si `y > BaseEmision`, entonces el ACK está confirmando uno o más de los segmentos no reconocidos anteriores. Así, el emisor actualiza su variable `BaseEmision` y reinicia el temporizador si actualmente aun existen segmentos no reconocidos.

Algunos escenarios interesantes

Acabamos de describir una versión enormemente simplificada de cómo TCP proporciona un servicio de transferencia de datos fiable. Pero incluso esta versión simplificada tiene sus sutilezas. Con el fin de clarificar cómo funciona este protocolo, vamos a analizar algunos escenarios interesantes. La Figura 3.34 describe el primero de estos escenarios, en el que el host A envía un segmento al host B. Suponga que ese segmento tiene el número de secuencia 92 y contiene 8 bytes de datos. Después de enviar este segmento, el host A espera un segmento procedente de B con un número de reconocimiento de 100. Aunque el segmento de A se recibe en B, el paquete de reconocimiento de B a A se pierde. En este caso, se produce un suceso de fin de temporización y el host A retransmite el mismo seg-

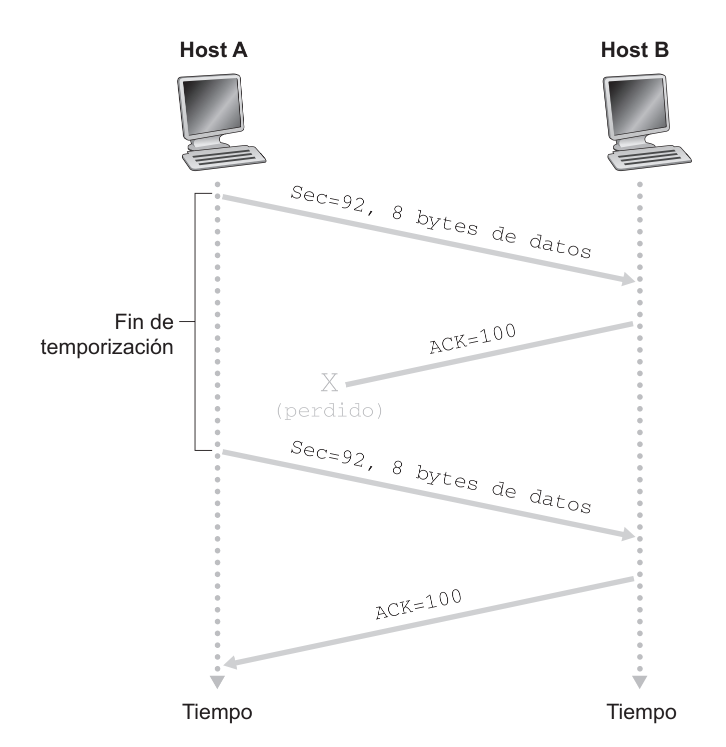


Figura 3.34 • Retransmisión debida a la pérdida de un paquete de reconocimiento ACK.

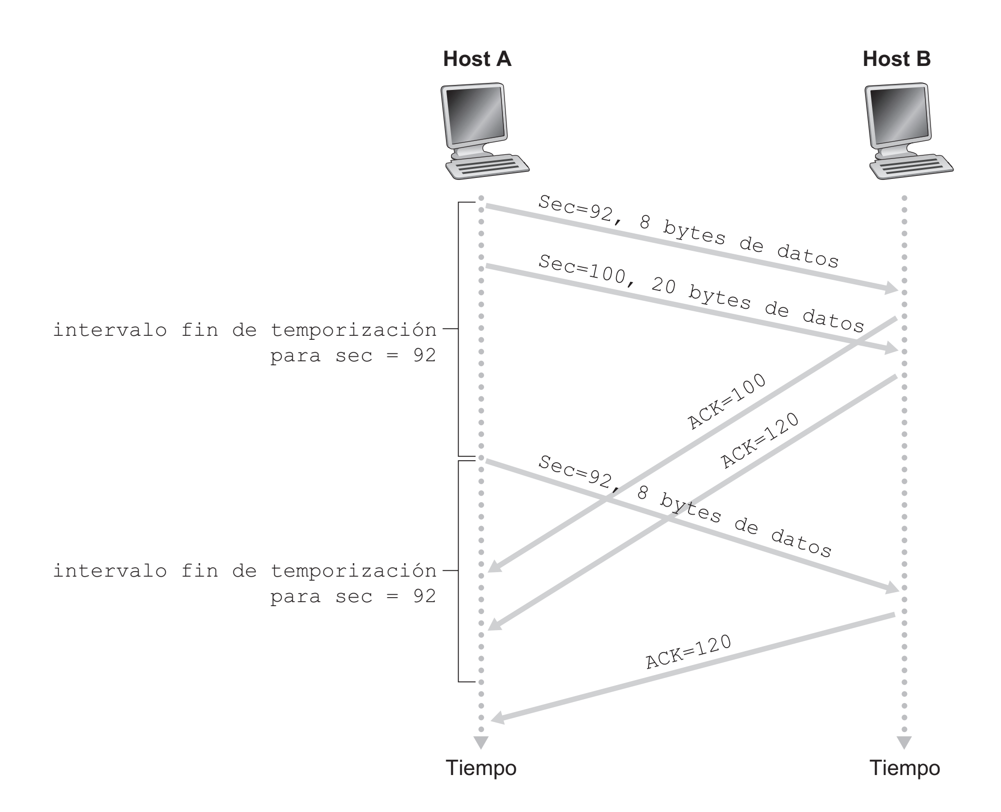


Figura 3.35 • Segmento 100 no retransmitido.

mento. Por supuesto, cuando el host B recibe la retransmisión, comprueba que de acuerdo con el número de secuencia el segmento contiene datos que ya habían sido recibidos. Por tanto, TCP en el host B descartará los bytes del segmento retransmitido.

En el segundo escenario, mostrado en la Figura 3.35, el host A envía dos segmentos seguidos. El primer segmento tiene el número de secuencia 92 y 8 bytes de datos, y el segundo segmento tiene el número de secuencia 100 y 20 bytes de datos. Suponga que ambos segmentos llegan intactos a B, y que B envía dos mensajes de reconocimiento separados para cada uno de estos segmentos. El primero de estos mensajes tiene el número de reconocimiento 100 y el segundo el número 120. Suponga ahora que ninguno de estos mensajes llega al host A antes de que tenga lugar el fin de la temporización. Cuando se produce el fin de temporización, el host A reenvía el primer segmento con el número de secuencia 92 y reinicia el temporizador. Siempre y cuando el ACK correspondiente al segundo segmento llegue antes de que tenga lugar un nuevo fin de temporización, el segundo segmento no se retransmitirá.

En el tercer y último escenario, suponemos que el host A envía los dos segmentos del mismo modo que en el segundo ejemplo. El paquete de reconocimiento del primer segmento se pierde en la red, pero justo antes de que se produzca el fin de la temporización, el host A recibe un paquete de reconocimiento con el número de reconocimiento 120. Por tanto, el host A sabe que el host B ha recibido *todo* hasta el byte 119; por tanto, el host A no reenvía ninguno de los dos segmentos. Este escenario se ilustra en la Figura 3.36.

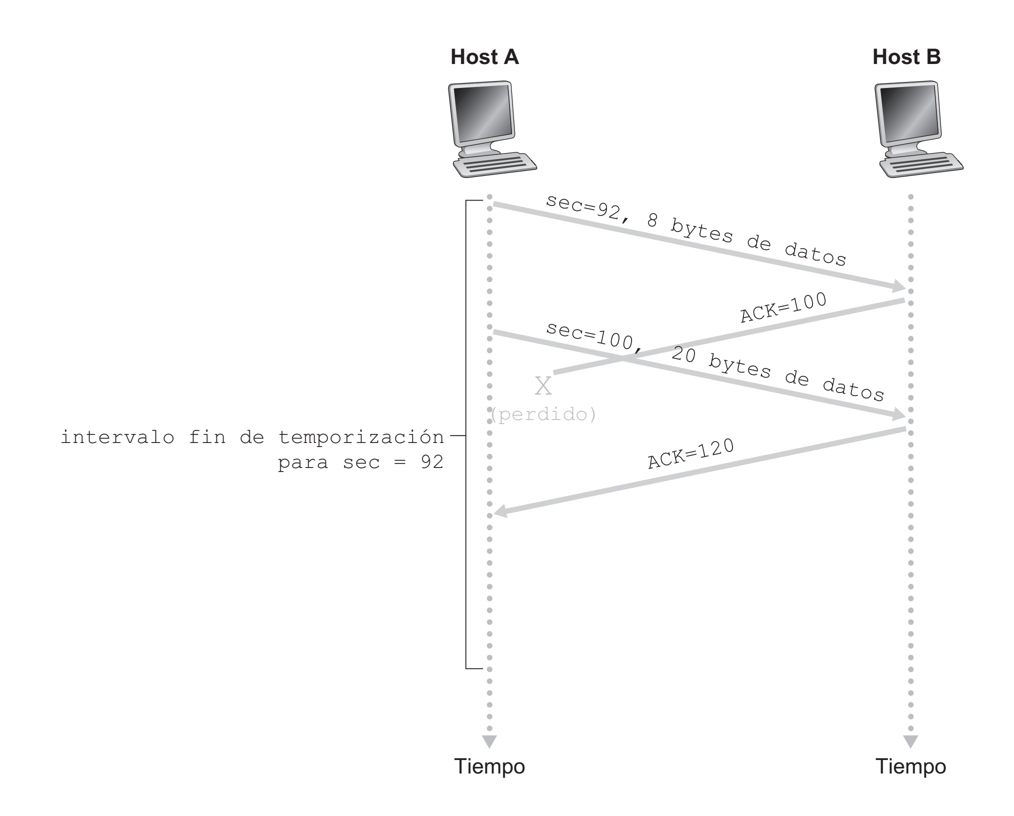


Figura 3.36 • Un reconocimiento acumulativo evita la retransmisión del primer segmento.

Duplicación del intervalo de fin de temporización

Examinemos ahora algunas de las modificaciones que aplican la mayor parte de las implementaciones TCP. La primera de ellas está relacionada con la duración del intervalo de fin de temporización después de que el temporizador ha caducado. En esta modificación, cuando tiene lugar un suceso de fin de temporización TCP retransmite el segmento aun no reconocido con el número de secuencia más pequeño, como se ha descrito anteriormente. Pero cada vez que TCP retransmite, define el siguiente intervalo de fin de temporización como dos veces el valor anterior, en lugar de obtenerlo a partir de los últimos valores de `RTTEstimado` y `RTTDesv` (como se ha descrito en la Sección 3.5.3). Por ejemplo, suponga que el `IntervaloFindeTemporización` asociado con el segmento no reconocido más antiguo es 0,75 segundos cuando el temporizador caduca por primera vez. Entonces TCP retransmitirá este segmento y establecerá el nuevo intervalo en 1,5 segundos. Si el temporizador caduca de nuevo 1,5 segundos más tarde, TCP volverá a retransmitir este segmento, estableciendo el intervalo de caducidad en 3,0 segundos. Por tanto, los intervalos crecen exponencialmente después de cada retransmisión. Sin embargo, cuando el temporizador se inicia después de cualquiera de los otros dos sucesos (es decir, datos recibidos de la aplicación y recepción de un ACK), el `IntervaloFindeTemporización` se obtiene a partir de los valores más recientes de `RTTEstimado` y `RTTDesv`.

Esta modificación proporciona una forma limitada de control de congestión (en la Sección 3.7 estudiaremos formas más exhaustivas de realizar el control de congestión en TCP). La caducidad del temporizador está causada muy probablemente por la congestión en la red, es decir, llegan demasiados paquetes a una (o más) colas de router a lo largo de la ruta entre el origen y el destino, haciendo que los paquetes sean descartados y/o sufran largos retardos de cola. Cuando existe congestión, si los orígenes continúan retransmitiendo paquetes persistentemente, la congestión puede empeorar. En lugar de ello, TCP actúa de forma más diplomática, haciendo que los emisores retransmitan después de intervalos cada vez más grandes. Cuando estudiemos CSMA/CD en el Capítulo 5, veremos que una idea similar se emplea en Ethernet .

Retransmisión rápida

Uno de los problemas con las retransmisiones generadas por los sucesos de fin de temporización es que el periodo de fin de temporización puede ser relativamente largo. Cuando se pierde un segmento, un periodo de fin de temporización largo fuerza al emisor a retardar el reenvío del paquete perdido, aumentando el retardo terminal a terminal. Afortunadamente, el emisor puede a menudo detectar la pérdida de paquetes antes de que tenga lugar el suceso de fin de temporización, observando los ACK duplicados. Un **ACK duplicado** es un ACK que vuelve a reconocer un segmento para el que el emisor ya ha recibido un reconocimiento anterior. Para comprender la respuesta del emisor a un ACK duplicado, tenemos que entender en primer lugar por qué el receptor envía un ACK duplicado. La Tabla 3.2 resume la política de generación de mensajes ACK en el receptor TCP [RFC 1122, RFC 2581]. Cuando un receptor TCP recibe un segmento con un número de secuencia que es mayor que el siguiente número de secuencia en orden esperado, detecta un hueco en el flujo de datos, es decir, detecta que falta un segmento. Este hueco podría ser el resultado de segmentos perdidos o reordenados dentro de la red. Dado que TCP no utiliza paquetes NAK, el receptor no puede devolver al emisor un mensaje de reconocimiento negativo explícito. En su lugar, simplemente vuelve a reconocer (es decir, genera un ACK duplicado) al último byte de datos

Suceso	Acción del receptor TCP
Llegada de un segmento en orden con el número de secuencia esperado. Todos los datos hasta el número de secuencia esperado ya han sido reconocidos.	ACK retardado. Esperar hasta durante 500 milisegundos la llegada de otro segmento en orden. Si el siguiente segmento en orden no llega en este intervalo, enviar un ACK.
Llegada de un segmento en orden con el número de secuencia esperado. Hay otro segmento en orden esperando la transmisión de un ACK.	Enviar inmediatamente un único ACK acumulativo, reconociendo ambos segmentos ordenados.
Llegada de un segmento desordenado con un número de secuencia más alto que el esperado. Se detecta un hueco.	Enviar inmediatamente un ACK duplicado, indicando el número de secuencia del siguiente byte esperado (que es el límite inferior del hueco).
Llegada de un segmento que completa parcial o completamente el hueco existente en los datos recibidos.	Enviar inmediatamente un ACK, suponiendo que el segmento comienza en el límite inferior del hueco.

Tabla 3.2 • Recomendación para la generación de mensajes ACK en TCP [RFC 1122, RFC 2581]

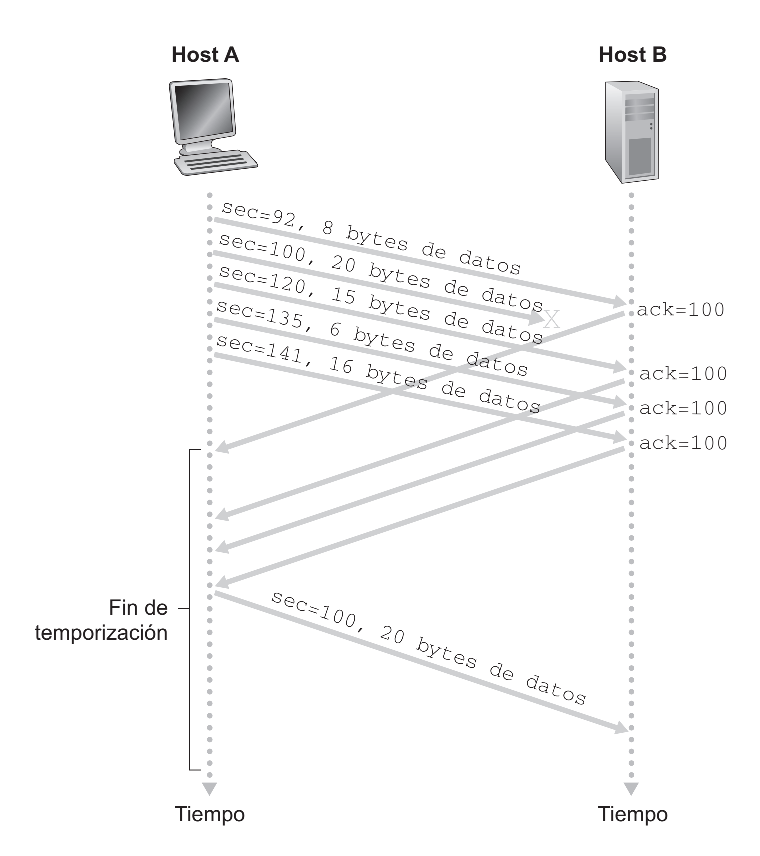


Figura 3.37 • Retransmisión rápida: retransmisión del segmento que falta antes de que caduque el temporizador del segmento.

en orden que ha recibido. Observe que la Tabla 3.2 contempla el caso de que el receptor no descarte los segmentos que no llegan en orden.

Puesto que un emisor suele enviar una gran número de segmentos seguidos, si se pierde uno de ellos, probablemente habrá muchos ACK duplicados seguidos. Si el emisor TCP recibe tres ACK duplicados para los mismos datos, toma esto como una indicación de que el segmento que sigue al segmento que ha sido reconocido tres veces se ha perdido (en los problemas de repaso, consideraremos la cuestión de por qué el emisor espera tres ACK duplicados, en lugar de un único duplicado). En el caso de que se reciban tres ACK duplicados, el emisor TCP realiza una **retransmisión rápida** [RFC 2581], reenviando el segmento que falta *antes* de que caduque el temporizador de dicho segmento. Esto se muestra en la Figura 3.37, en la que se pierde el segundo segmento y luego se retransmite antes de que caduque su temporizador. Para TCP con retransmisión rápida, el siguiente fragmento de código reemplaza al suceso de recepción de un ACK de la Figura 3.33:

```
suceso: ACK recibido, con un valor de campo ACK de y
            if (y > BaseEmision) {
                  BaseEmision = y
                  if (existen segmentos pendientes de reconocimiento)
                       iniciar temporizador
```

```
    }
  else { /* un ACK duplicado para un segmento ya reconocido */
     incrementar el número de mensajes ACK duplicados
        recibidos para y
     if (número de ACK duplicados recibidos para y==3)
        /* TCP con retransmisión rápida */
        reenviar el segmento con número de secuencia y
     }
            break;
```

Anteriormente hemos mencionado que surgen muchos problemas sutiles cuando se implementa un mecanismo de fin de temporización/retransmisión en un protocolo real tal como TCP. Los procedimientos anteriores, que se han desarrollado como resultado de más de 15 años de experiencia con los temporizadores TCP, deberían convencerle de hasta que punto son sutiles esos problemas.

Retroceder N o Repetición selectiva

Profundicemos algo más en nuestro estudio del mecanismo de recuperación de errores de TCP considerando la siguiente cuestión: ¿Es TCP un protocolo GBN o un protocolo SR? Recuerde que los reconocimientos de TCP son acumulativos y que los segmentos correctamente recibidos pero no en orden no son reconocidos individualmente por el receptor. En consecuencia, como se muestra en la Figura 3.33 (véase también la Figura 3.19), el emisor TCP sólo necesita mantener el número de secuencia más pequeño de un byte transmitido pero no reconocido (`BaseEmision`) y el número de secuencia del siguiente byte que va a enviar (`SigNumSec`). En este sentido, TCP se parece mucho a un protocolo de tipo GBN. No obstante, existen algunas diferencia entre TCP y Retroceder N. Muchas implementaciones de TCP almacenan en buffer los segmentos recibidos correctamente pero no en orden [Stevens 1994]. Considere también lo que ocurre cuando el emisor envía una secuencia de segmentos 1, 2, . . . , N, y todos ellos llegan en orden y sin errores al receptor. Suponga además que el paquete de reconocimiento para el paquete $n < N$ se pierde, pero los $N - 1$ paquetes de reconocimiento restantes llegan al emisor antes de que tengan lugar sus respectivos fines de temporización. En este ejemplo, GBN retransmitiría no sólo el paquete n, sino también todos los paquetes subsiguientes $n + 1$, $n + 2$, . . . , N. Por otro lado, TCP retransmitiría como mucho un segmento, el segmento n. Además, TCP no retransmitiría ni siquiera el segmento n si el reconocimiento para el segmento $n + 1$ llega antes del fin de temporización correspondiente al segmento n.

Una modificación propuesta para TCP es lo que se denomina **reconocimiento selectivo** [RFC 2018], que permite a un receptor TCP reconocer segmentos no ordenados de forma selectiva, en lugar de sólo hacer reconocimientos acumulativos del último segmento recibido correctamente y en orden. Cuando se combina con la retransmisión selectiva (saltándose la retransmisión de segmentos que ya han sido reconocidos de forma selectiva por el receptor), TCP se comporta como nuestro protocolo SR selectivo. Por tanto, el mecanismo de recuperación de errores de TCP probablemente es mejor considerarlo como un híbrido de los protocolos GBN y SR.

3.5.5 Control de flujo

Recuerde que los hosts situados a cada lado de una conexión TCP disponen de un buffer de recepción para la conexión. Cuando la conexión TCP recibe bytes que son correctos y en

secuencia, coloca los datos en el buffer de recepción. El proceso de aplicación asociado leerá los datos de este buffer, pero no necesariamente en el instante en que llegan. De hecho, la aplicación receptora puede estar ocupada con alguna otra tarea y puede incluso no leer los datos hasta mucho tiempo después de que estos hayan llegado. Si la aplicación es relativamente lenta en lo que respecta a la lectura de los datos, el emisor puede fácilmente desbordar el buffer de recepción de la conexión enviando muchos datos demasiado rápidamente.

TCP proporciona un **servicio de control de flujo** a sus aplicaciones para eliminar la posibilidad de que el emisor desborde el buffer del receptor. El control de flujo es por tanto un servicio de adaptación de velocidades (adapta la velocidad a la que el emisor está transmitiendo frente a la velocidad a la que la aplicación receptora está leyendo). Como hemos mencionado anteriormente, un emisor TCP también puede atascarse debido a la congestión de la red IP; esta forma de control del emisor se define como un mecanismo de **control de congestión**, un tema que exploraremos en detalle en las Secciones 3.6 y 3.7. Aunque las acciones tomadas por los controles de flujo y de congestión son similares (regular el flujo del emisor), obviamente se toman por razones diferentes. Lamentablemente, muchos autores utilizan los términos de forma indistinta, por lo que los lectores conocedores del tema deberían tratar de diferenciarlos. Examinemos ahora cómo proporciona TCP su servicio de control de flujo. En esta sección vamos a suponer que la implementación de TCP es tal que el receptor TCP descarta los segmentos que no llegan en orden.

TCP proporciona un servicio de control de flujo teniendo que mantener el *emisor* una variable conocida como **ventana de recepción**. Informalmente, la ventana de recepción se emplea para proporcionar al emisor una idea de cuánto espacio libre hay disponible en el buffer del receptor. Puesto que TCP es una conexión full-duplex, el emisor de cada lado de la conexión mantiene una ventana de recepción diferente. Estudiemos la ventana de recepción en el contexto de una operación de transferencia de un archivo. Suponga que el host A está enviando un archivo grande al host B a través de una conexión TCP. El host B asigna un buffer de recepción a esta conexión; designamos al tamaño de este buffer como `BufferRecepción`. De vez en cuando, el proceso de aplicación del host B lee el contenido del buffer. Definimos las siguientes variables:

- `UltimoByteLeido`: el número del último byte del flujo de datos del buffer leído por el proceso de aplicación del host B.
- `UltimoByteRecibido`: el número del último byte del flujo de datos que ha llegado procedente de la red y que se ha almacenado en el buffer de recepción del host B.

Puesto que en TCP no está permitido desbordar el buffer asignado, tenemos que:

`UltimoByteRecibido – UltimoByteLeido ≤ BufferRecepcion`

La ventana de recepción se hace igual a la cantidad de espacio libre disponible en el buffer:

`VentanaRecepcion = BufferRecepcion – [UltimoByteRecibido – UltimoByteLeido]`

Dado que el espacio libre varía con el tiempo, `VentanaRecepcion` es una variable dinámica, la cual se ilustra en la Figura 3.38.

¿Cómo utiliza la conexión la variable `VentanaRecepcion` para proporcionar el servicio de control de flujo? El host B dice al host A la cantidad de espacio disponible que hay en el buffer de la conexión almacenando el valor actual de `VentanaRecepcion` en el campo ventana de recepción de cada segmento que envía a A. Inicialmente, el host B establece que

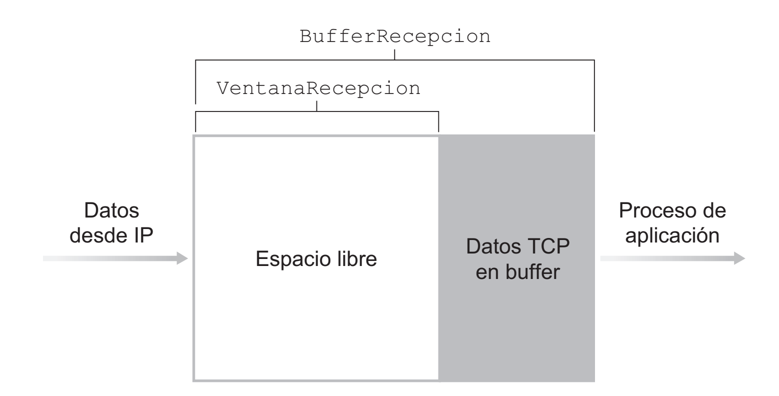


Figura 3.38 • La ventana de recepción y el buffer de recepción (`BufferRecepcion`).

`VentanaRecepcion` = `BufferRecepcion`. Observe que para poder implementar este mecanismo, el host B tiene que controlar diversas variables específicas de la conexión.

A su vez, el host A controla dos variables, `UltimoByteEnviado` y `UltimoByteReconocido`, cuyos significados son obvios. Observe que la diferencia entre estas dos variables, `UltimoByteEnviado – UltimoByteReconocido`, es la cantidad de datos no reconocidos que el host A ha enviado a través de la conexión. Haciendo que el número de datos no reconocidos sea inferior al valor de `VentanaRecepcion`, el host A podrá asegurarse de no estar desbordando el buffer de recepción en el host B. Por tanto, el host A se asegura, a todo lo largo del tiempo de vida de la conexión, de que:

`UltimoByteEnviado – UltimoByteReconocido` ≤ `VentanaRecepcion`

Existe un pequeño problema técnico con este esquema. Para ver de qué se trata, suponga que el buffer de recepción del host B está lleno, de manera que `VentanaRecepcion` = 0. Después de anunciar al host A que `VentanaRecepcion` = 0, suponga también que B no tiene *nada* que enviar a A. Veamos qué es lo que ocurre. A medida que el proceso de aplicación del host B vacía el buffer, TCP no envía nuevos segmentos con valores nuevos `VentanaRecepcion` al host A; por supuesto, TCP envía un segmento al host A sólo si tiene datos que enviar o si tiene que enviar un paquete de reconocimiento. Por tanto, el host A nunca es informado de que hay algo de espacio en el buffer de recepción del host B (el host A está bloqueado y no puede transmitir más datos). Para resolver este problema, la especificación TCP requiere al host A que continúe enviando segmentos con un byte de datos cuando la longitud de la ventana de recepción de B es cero. Estos segmentos serán reconocidos por el receptor. Finalmente, el buffer comenzará a vaciarse y los ACK contendrán un valor de `VentanaRecepcion` distinto de cero.

El sitio web del libro en http://www.awl.com/kurose-ross proporciona un applet Java interactivo que ilustra el funcionamiento de la ventana de recepción de TCP.

Debemos comentar, una vez que hemos descrito el servicio de control de flujo de TCP, que UDP no proporciona ningún mecanismo de control de flujo. Para entender esta cuestión, consideremos la transmisión de una serie de segmentos UDP desde un proceso que se ejecuta en el host A a un proceso que se ejecuta en el host B. En una implementación típica de UDP, el protocolo UDP almacenará los segmentos en un buffer de tamaño finito que “pre-

cede" al correspondiente socket (es decir, la puerta de entrada al proceso). El proceso lee un segmento completo del buffer cada vez. Si el proceso no lee los segmentos del buffer lo suficientemente rápido, éste se desbordará y los segmentos serán descartados.

3.5.6 Gestión de la conexión TCP

En esta subsección vamos a ver cómo se establece y termina una conexión TCP. Aunque este tema no parece particularmente emocionante, tiene una gran importancia, porque el establecimiento de una conexión TCP puede aumentar significativamente el retardo percibido (por ejemplo, cuando se navega por la Web). Además, muchos de los ataques de red más comunes, incluyendo el increíblemente popular ataque por inundación SYN, explotan las vulnerabilidades de la gestión de una conexión TCP. En primer lugar, veamos cómo se establece una conexión TCP. Suponga que hay un proceso en ejecución en un host (cliente) que desea iniciar una conexión con otro proceso que se ejecuta en otro host (servidor). El proceso de aplicación cliente informa en primer lugar al cliente TCP que desea establecer una conexión con un proceso servidor. A continuación, el protocolo TCP en el cliente establece una conexión TCP con el protocolo TCP en el servidor de la siguiente manera:

- *Paso 1*. En primer lugar, TCP del lado del cliente envía un segmento TCP especial al TCP del lado servidor. Este segmento especial no contiene datos de la capa de aplicación. Pero uno de los bits indicadores de la cabecera del segmento (véase la Figura 3.29), el bit SYN, se pone a 1. Por esta razón, este segmento especial se referencia como un segmento SYN. Además, el cliente selecciona de forma aleatoria un número de secuencia inicial (`cliente_nsi`) y lo coloca en el campo número de secuencia del segmento TCP inicial SYN. Este segmento se encapsula dentro de un datagrama IP y se envía al servidor. Es importante que esta elección aleatoria del valor de `cliente_nsi` se haga apropiadamente con el fin de evitar ciertos ataques de seguridad [CERT 2001-09].
- *Paso 2*. Una vez que el datagrama IP que contiene el segmento SYN TCP llega al host servidor (¡suponiendo que llega!), el servidor extrae dicho segmento SYN del datagrama, asigna los buffers y variables TCP a la conexión y envía un segmento de conexión concedida al cliente TCP. (Veremos en el Capítulo 8 que la asignación de estos buffers y variables antes de completar el tercer paso del proceso de acuerdo en tres fases hace que TCP sea vulnerable a un ataque de denegación de servicio, conocido como ataque por inundación SYN.) Este segmento de conexión concedida tampoco contiene datos de la capa de aplicación. Sin embargo, contiene tres fragmentos de información importantes de la cabecera del segmento. El primero, el bit SYN se pone a 1. El segundo, el campo reconocimiento de la cabecera del segmento TCP se hace igual a `cliente_nsi+1`. Por último, el servidor elige su propio número de secuencia inicial (`servidor_nsi`) y almacena este valor en el campo número de secuencia de la cabecera del segmento TCP. Este segmento de conexión concedida está diciendo, en efecto, "He recibido tu paquete SYN para iniciar una conexión con tu número de secuencia inicial, `cliente_nsi`. Estoy de acuerdo con establecer esta conexión. Mi número de secuencia inicial es `servidor_nsi`". El segmento de conexión concedida se conoce como **segmento SYNACK**.
- *Paso 3*. Al recibir el segmento SYNACK, el cliente también asigna buffers y variables a la conexión. El host cliente envía entonces al servidor otro segmento; este último segmento confirma el segmento de conexión concedida del servidor (el cliente hace esto

almacenando el valor `servidor_nsi+1` en el campo de reconocimiento de la cabecera del segmento TCP). El bit SYN se pone a cero, ya que la conexión está establecida. Esta tercera etapa del proceso de acuerdo en tres fases puede transportar datos del cliente al servidor dentra de la carga útil del segmento.

Una vez completados estos tres pasos, los hosts cliente y servidor pueden enviarse segmentos que contengan datos el uno al otro. En cada uno de estos segmentos futuros, el valor del bit SYN será cero. Observe que con el fin de establecer la conexión se envían tres paquetes entre los dos hosts, como se ilustra en la Figura 3.39. Por ello, este procedimiento de establecimiento de la conexión suele denominarse proceso de **acuerdo en tres fases**. En los problemas de repaso se exploran varios aspectos de este proceso de TCP (¿Por qué se necesitan los números de secuencia iniciales? ¿Por qué se necesita un proceso de acuerdo en tres fases en lugar de uno en dos fases?). Es interesante observar que un escalador y la persona que le asegura (que se encuentra por debajo del escalador y cuya tarea es sostener la cuerda de seguridad del mismo) utilizan un protocolo de comunicaciones con un proceso de acuerdo en tres fases que es idéntico al de TCP, para garantizar que ambas partes estén preparadas antes de que el escalador inicie el ascenso.

Todo lo bueno se termina y esto es aplicable también a una conexión TCP. Cualquiera de los dos procesos participantes en una conexión TCP pueden dar por teminada dicha conexión. Cuando una conexión se termina, los "recursos" (es decir, los buffers y las variables) de los hosts se liberan. Por ejemplo, suponga que el cliente decide cerrar la conexión, como se muestra en la Figura 3.40. El proceso de la aplicación cliente ejecuta un comando de cierre. Esto hace que el cliente TCP envíe un segmento especial TCP al proceso servidor. El

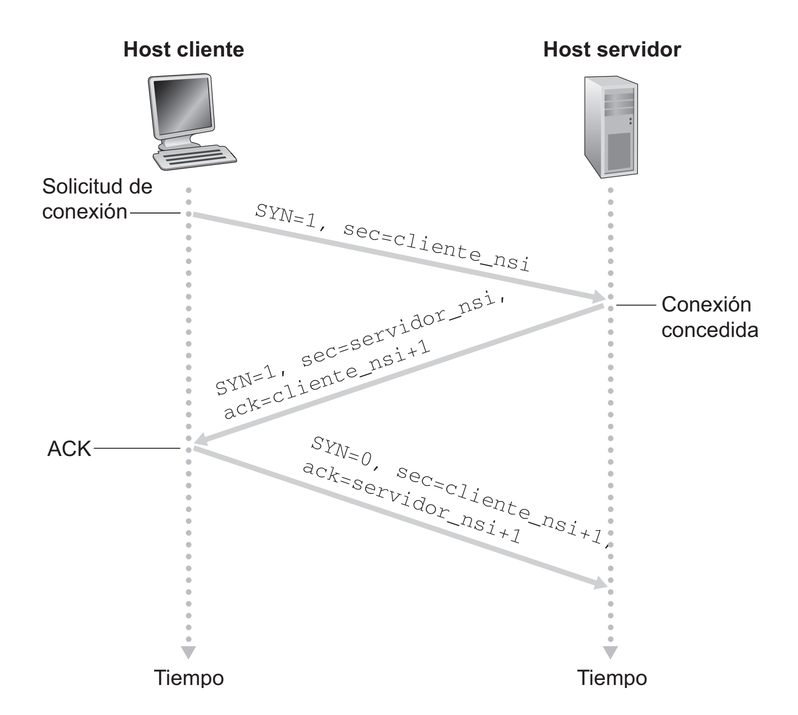


Figura 3.39 • El proceso de acuerdo en tres fases de TCP: intercambio de segmentos.

segmento especial contiene un bit indicador en la cabecera del segmento, el bit FIN (véase la Figura 3.29), puesto a 1. Cuando el servidor recibe este segmento, devuelve al cliente un segmento de reconocimiento. El servidor entonces envía su propio segmento de desconexión, que tiene el bit FIN puesto a 1. Por último, el cliente reconoce el segmento de desconexión del servidor. En esta situación, los recursos de ambos hosts quedan liberados.

Mientras se mantiene una conexión TCP, el protocolo TCP que se ejecuta en cada host realiza transiciones a través de los diversos **estados TCP**. La Figura 3.41 ilustra una secuencia típica de los estados TCP visitados por el *cliente* TCP, el cual se inicia en el estado CLOSED (cerrado). La aplicación en el lado del cliente inicia una nueva conexión (creando un objeto Socket en nuestros ejemplos de Java del Capítulo 2). Esto hace que TCP en el cliente envíe un segmento SYN a TCP en el servidor. Después de haber enviado el segmento SYN, el cliente TCP entra en el estado SYN_SENT (SYN_enviado). Mientras se encuentra en este estado, el cliente TCP espera un segmento procedente del TCP servidor que incluya un reconocimiento del segmento anterior del cliente y que tenga el bit SYN puesto a 1. Una vez recibido ese segmento, el cliente TCP entra en el estado ESTABLISHED (establecido). Mientras está en este último estado, el cliente TCP puede enviar y recibir segmentos TCP que contengan datos de carga útil (es decir, datos generados por la aplicación).

Suponga que la aplicación cliente decide cerrar la conexión (tenga en cuenta que el servidor también podría decidir cerrar la conexión). Así, el cliente TCP envía un segmento TCP con el bit FIN puesto a 1 y entra en el estado FIN_WAIT_1 (FIN_espera_1). En este estado, el cliente TCP queda a la espera de un segmento TCP procedente del servidor que contenga un mensaje de reconocimiento. Cuando lo recibe, el cliente TCP pasa al estado

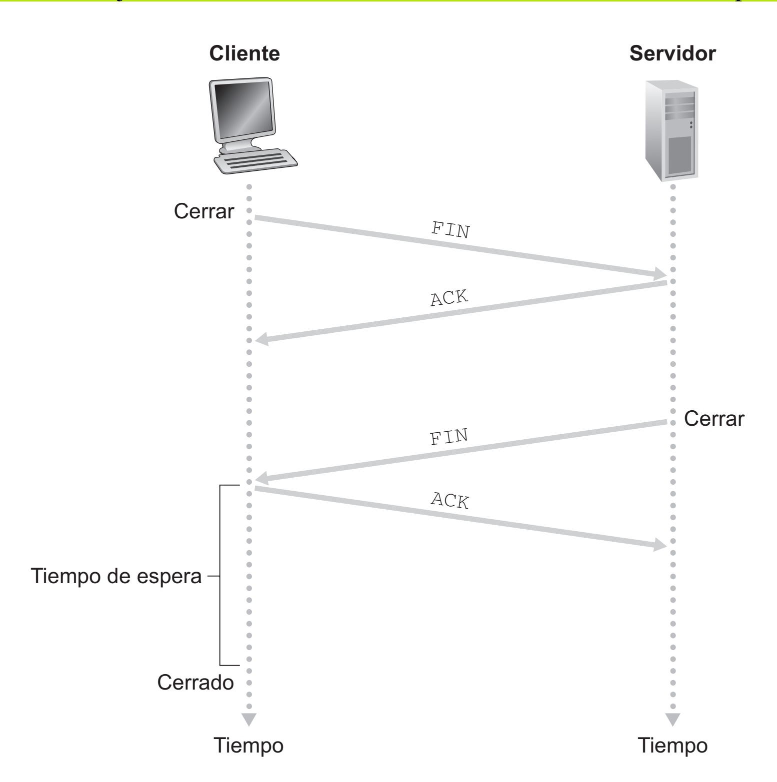


Figura 3.40 • Cierre de una conexión TCP.

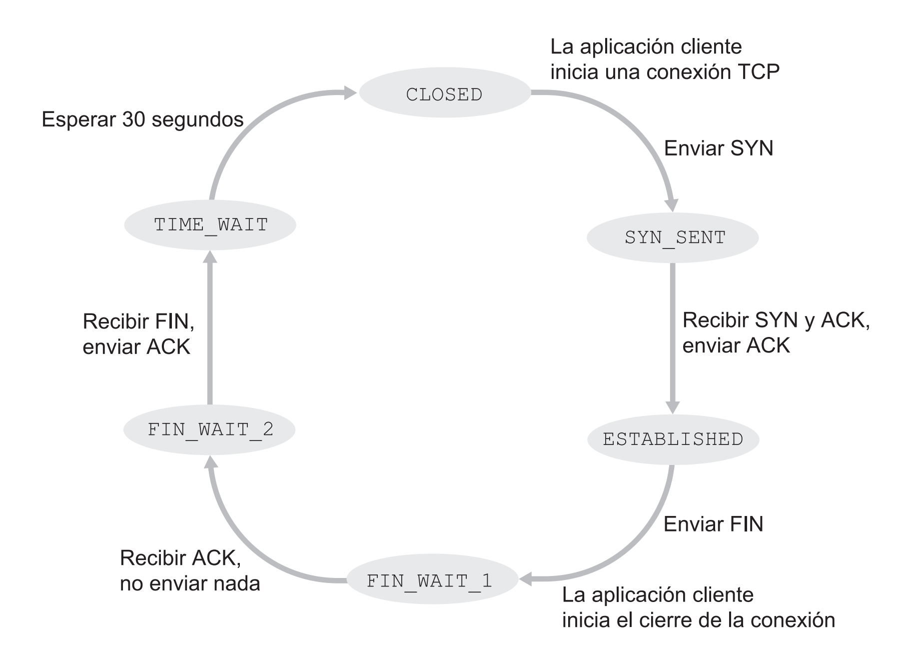


Figura 3.41 • Secuencia típica de estados TCP visitados por un cliente TCP.

FIN_WAIT_2. En este estado, el cliente espera a recibir otro segmento del servidor con el bit FIN puesto a 1; una vez recibido, el cliente TCP reconoce el segmento del servidor y pasa al estado TIME_WAIT, en el que puede reenviar al cliente TCP el reconocimiento final en caso de que el paquete ACK se pierda. El tiempo invertido en el estado TIME_WAIT es dependiente de la aplicación, siendo sus valores típicos 30 segundos, 1 minuto y 2 minutos. Después de este tiempo de espera, la conexión se cierra y todos los recursos del lado del cliente (incluyendo los números de puerto) son liberados.

La Figura 3.42 ilustra la serie de estados que visita normalmente el TCP del lado del servidor, suponiendo que el cliente inicia el cierre de la conexión. Las transiciones se explican por sí mismas. En estos dos diagramas de transiciones de estados únicamente hemos mostrado cómo se establece y termina normalmente una conexión TCP. No hemos descrito lo que ocurre en determinados escenarios problemáticos; por ejemplo, cuando ambos lados de una conexión desean iniciar o terminar al mismo tiempo la conexión. Si está interesado en este tema y en otros algo más avanzados sobre TCP, le animamos a consultar el exhaustivo libro de [Stevens 1994].

Hasta aquí, hemos supuesto que tanto el cliente como el servidor están preparados para comunicarse; es decir, que el servidor está escuchando en el puerto al que el cliente envía su segmento SYN. Consideremos lo que ocurre cuando un host recibe un segmento TCP cuyo número de puerto o cuya dirección IP de origen no se corresponde con ninguno de los sockets activos en el host. Por ejemplo, suponga que un host recibe un paquete TCP SYN cuyo puerto de destino es el número 80, pero el host no está aceptando conexiones en dicho puerto (es decir, no está ejecutando un servidor web en el puerto 80). Entonces, el host enviará al origen un segmento especial de reinicio. Este segmento TCP tiene el bit indicador RST (véase la Sección 3.5.2) puesto a 1. Por tanto, cuando un host envía un segmento de reinicio,

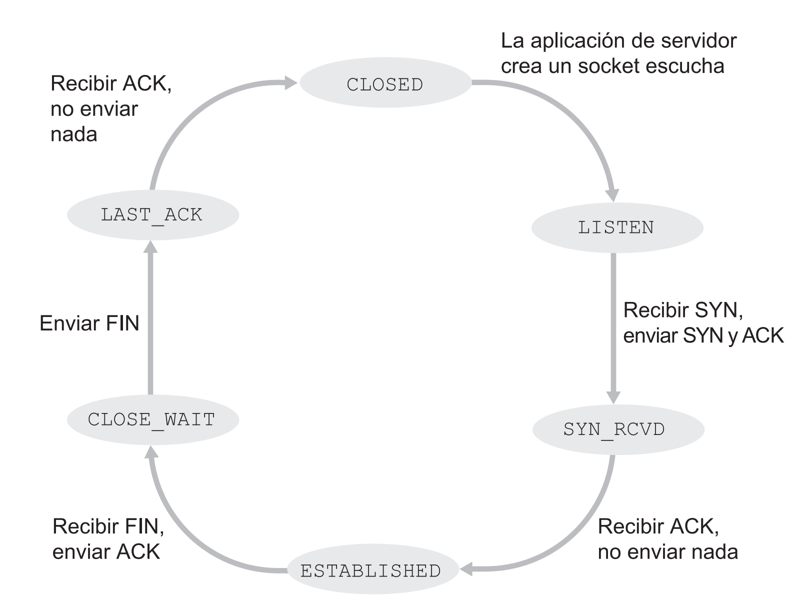


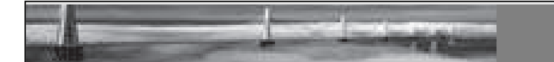
Figura 3.42 • Secuencia típica de estados TCP visitados por un servidor TCP.

le está diciendo al emisor "No tengo un socket para ese segmento. Por favor, no reenvies el segmento." Cuando un host recibe un paquete UDP en el que el número de puerto de destino no se corresponde con un socket UDP activo, el host envía un datagrama ICMP especial, como se explica en el Capítulo 4.

Ahora que ya tenemos unos buenos conocimientos sobre cómo se gestiona una conexión TCP, vamos a volver sobre la herramienta de exploración de puertos `nmap` y vamos a ver más detalladamente cómo funciona. Para explorar un puerto TCP específico, por ejemplo, el puerto 6789, en un host objetivo, `nmap` enviará un segmento TCP SYN con el puerto de destino 6789 a dicho host. Pueden obtenerse entonces tres posibles resultados:

- *El host de origen recibe un segmento TCP SYNACK del host objetivo*. Dado que esto significa que hay una aplicación ejecutándose con TCP en el puerto 6789 del host objetivo, `nmap` devuelve "open" (puerto abierto).
- *El host de origen recibe un segmento TCP RST procedente desde el host objetivo*. Esto significa que el segmento SYN ha alcanzado el host objetivo, pero éste no está ejecutando una aplicación con TCP en el puerto 6789. Pero el atacante sabrá como mínimo que el segmento destinado al puerto 6789 del host no está bloqueado por ningún cortafuegos existente en la ruta entre los hosts de origen y de destino. (Los cortafuegos se estudian en el Capítulo 8.)
- *El origen no recibe nada*. Probablemente, esto significa que el segmento SYN fue bloqueado por un cortafuegos intermedio y nunca llegó al host objetivo.

`nmap` es una potente herramienta que puede detectar "las brechas en el muro", no sólo en lo relativo a los puertos TCP abiertos, sino también a los puertos UDP abiertos, a los cortafuegos y sus configuraciones e incluso, a las versiones de aplicaciones y sistemas operativos. La mayor parte de esta tarea se lleva a cabo manipulando los segmentos de gestión de



SEGURIDAD

EL ATAQUE POR INUNDACIÓN SYN

Hemos visto el proceso de acuerdo en tres fases de TCP en el que un servidor asigna e inicializa los buffers y variables de una conexión en respuesta a la recepción de un segmento SYN. A continuación, el servidor envía como respuesta un segmento SYNACK y espera al correspondiente segmento de reconocimiento (ACK) procedente del cliente, el tercer y último paso del proceso de acuerdo antes de que la conexión quede completamente establecida. Si el cliente no envía un segmento ACK para completar el tercero de los pasos del proceso de acuerdo en tres fases, al final (a menudo después de un minuto o más) el servidor terminará la conexión semiabierta y reclamará los recursos asignados.

Este protocolo de gestión de la conexión TCP establece la base para un ataque DoS clásico, conocido como **ataque por inundación SYN**. En este ataque, los malos envían un gran número de segmentos SYN TCP, sin completar el tercer paso del proceso de acuerdo. El ataque se puede amplificar enviando segmentos SYN desde varios orígenes, creando un ataque por inundación SYN DDoS (*Distributed Denial of Service*, Servicio de denegación distribuido). Con esta gran cantidad de segmentos SYN, los recursos de conexión del servidor pueden agotarse rápidamente a medida que se van asignando (¡aunque nunca se utilizan!) a conexiones semiabiertas. Con los recursos del servidor agotados, se niega el servicio a los clientes legítimos. Estos ataques por inundación SYN [CERT SYN 1996] se encuentran entre los primeros ataques DoS documentados por el CERT [CERT 2009].

La inundación con segmentos SYN es un ataque potencialmente devastador. Afortunadamente, existe una defensa efectiva, denominada **cookies SYN** [Skoudis 2006; Cisco SYN 2009; Bernstein 2009], actualmente implantada en la mayoría de los principales sistemas operativos. Las cookies SYN funcionan del siguiente modo:

- Cuando el servidor recibe un segmento SYN, no sabe si ese segmento procede de un usuario legítimo o forma parte de un ataque por inundación SYN. Por tanto, el servidor no crea una conexión semiabierta TCP para ese segmento SYN. En su lugar, el servidor crea un número de secuencia TCP inicial que es una función compleja (una función hash) de las direcciones IP de origen y de destino y los números de puerto del segmento SYN, así como de un número secreto que únicamente conoce el servidor. (El servidor utiliza el mismo número secreto para un gran número de conexiones.) Este número de secuencia inicial cuidadosamente confeccionado se denomina “cookie”. El servidor tiene entonces que enviar un paquete SYNACK con este número de secuencia inicial especial. *Es importante que el servidor no recuerde la cookie ni ninguna otra información de estado correspondiente al paquete SYN.*
- Si el cliente es legítimo, entonces devolverá un segmento ACK. El servidor, al recibir este ACK, tiene que verificar que corresponde a algún SYN enviado anteriormente. ¿Cómo se hace esto si el servidor no mantiene memoria acerca de los segmentos SYN? Como ya habrá imaginado, se hace utilizando la cookie. Concretamente, para un segmento ACK legítimo, el valor contenido en el campo de reconocimiento es igual al número de secuencia del segmento SYNACK más uno (véase la Figura 3.39). A continuación, el servidor ejecuta la misma función utilizando los mismos campos en el segmento ACK y

SEGURIDAD

el número secreto. Si el resultado de la función más uno es igual que el número de reconocimiento, el servidor concluye que el ACK se corresponde con un segmento SYN anterior y, por tanto, lo valida. A continuación, el servidor crea una conexión completamente abierta junto con un socket.

- Por el contrario, si el cliente no devuelve un segmento ACK, entonces el segmento SYN original no causa ningún daño al servidor, ya que éste no le ha asignado ningún recurso.

Las cookies SYN eliminan de forma efectiva la amenaza de un ataque por inundación SYN. Una variación de esta clase de ataque es que el cliente malicioso devuelva un segmento ACK válido para cada segmento SYNACK que el servidor genera. Esto hará que el servidor establezca conexiones TCP *completamente abiertas*, incluso aunque su sistema operativo emplee cookies SYN. Si decenas de miles de clientes están siendo utilizados (ataque DDoS), teniendo cada uno una dirección IP distinta, entonces será difícil para el servidor diferenciar entre los orígenes legítimos y los maliciosos. Por tanto, contra este "ataque que completa el proceso de acuerdo" puede ser más difícil defenderse que contra un ataque clásico por inundación SYN.

la conexión TCP [Skoudis 2006]. Si se encuentra cerca de una máquina Linux, puede realizar algunas pruebas directamente con `nmap`, simplemente escribiendo "`nmap`" en la línea de comandos. Puede descargar `nmap` para otros sistemas operativos en la dirección http://insecure.org/nmap.

Con esto completamos nuestra introducción a los mecanismos de control de errores y de control de flujo de TCP. En la Sección 3.7, volveremos sobre TCP y estudiaremos más detalladamente el control de congestión de TCP. Sin embargo, antes de eso, vamos a dar un paso atrás y vamos a examinar los problemas de control de congestión en un contexto más amplio.

TP11

3.6 Principios del control de congestión

En la sección anterior hemos examinado tanto los principios generales como los específicos de los mecanismos de TCP utilizados para proporcionar un servicio de transferencia de datos fiable en lo que se refiere a la pérdida de paquetes. Anteriormente habíamos mencionado que, en la práctica, tales pérdidas son normalmente el resultado de un desbordamiento de los buffers de los routers a medida que la red se va congestionando. La retransmisión de paquetes por tanto se ocupa de un síntoma de la congestión de red (la pérdida de un segmento específico de la capa de transporte) pero no se ocupa de la causa de esa congestión de la red (demasiados emisores intentando transmitir datos a una velocidad demasiado alta). Para tratar la causa de la congestión de la red son necesarios mecanismos que regulen el flujo de los emisores en cuanto la congestión de red aparezca.

En esta sección consideraremos el problema del control de congestión en un contexto general, con el fin de comprender por qué la congestión es algo negativo, cómo la conges-

ción de la red se manifiesta en el rendimiento ofrecido a las aplicaciones de la capa superior y cómo pueden aplicarse diversos métodos para evitar la congestión de la red o reaccionar ante la misma. Este estudio de carácter general del control de la congestión es apropiado porque, puesto que junto con la transferencia de datos fiable, se encuentra al principio de la lista de los diez problemas más importantes de las redes. Concluiremos esta sección con una exposición acerca del control de congestión en el servicio **ABR (*Available Bit-Rate*, velocidad de bit disponible)** de las redes **ATM (*Asynchronous Transfer Mode*, Modo de transferencia asíncrono)**. En la siguiente sección se lleva a cabo un estudio detallado sobre el algoritmo de control de congestión de TCP.

3.6.1 Las causas y los costes de la congestión

Iniciemos este estudio sobre el control de congestión examinando tres escenarios con una complejidad creciente en los que se produce congestión. En cada caso, veremos en primer lugar por qué se produce la congestión y el coste de la misma (en términos de recursos no utilizados por completo y del bajo rendimiento ofrecido a los sistemas terminales). Todavía no vamos a entrar a ver cómo reaccionar ante una congestión, o cómo evitarla, simplemente vamos a poner el foco sobre la cuestión más simple de comprender: qué ocurre cuando los hosts incrementan su velocidad de transmisión y la red comienza a congestionarse.

Escenario 1: dos emisores, un router con buffers de capacidad ilimitada

Veamos el escenario de congestión más simple posible: dos hosts (A y B), cada uno de los cuales dispone de una conexión que comparte un único salto entre el origen y el destino, como se muestra en la Figura 3.43.

Supongamos que la aplicación del host A está enviando datos a la conexión (por ejemplo, está pasando datos al protocolo de la capa de transporte a través de un socket) a una velocidad media de λ_{in} bytes/segundo. Estos datos son originales en el sentido de que cada unidad de datos se envía sólo una vez al socket. El protocolo del nivel de transporte subyacente es un protocolo simple. Los datos se encapsulan y se envían; no existe un mecanismo

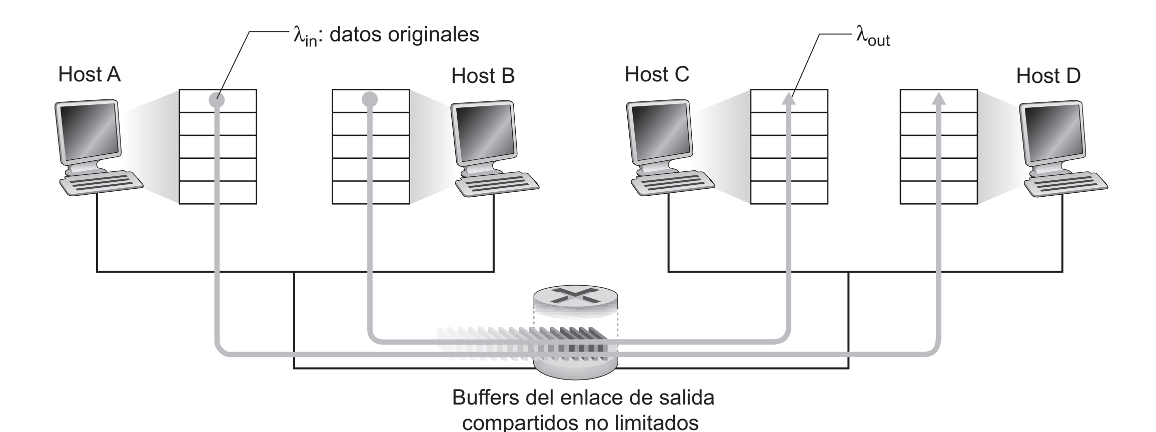


Figura 3.43 • Escenario de congestión 1: dos conexiones que comparten un único salto con buffers de capacidad ilimitada.

de recuperación de errores (como por ejemplo, las retransmisiones), ni se realiza un control de flujo ni un control de congestión. Ignorando la sobrecarga adicional debida a la adición de la información de cabecera de las capas de transporte e inferiores, la velocidad a la que el host A entrega tráfico al router en este primer escenario es por tanto λ_{in} bytes/segundo. El host B opera de forma similar, y suponemos, con el fin de simplificar, que también está enviando datos a una velocidad de λ_{in} bytes/segundo. Los paquetes que salen de los hosts A y B atraviesan un router y un enlace de salida compartido de capacidad *R*. El router tiene buffers que le permiten almacenar paquetes entrantes cuando la tasa de llegada de paquetes excede la capacidad del enlace de salida. En este primer escenario, suponemos que el router tiene un buffer con una cantidad de espacio infinita.

La Figura 3.44 muestra el rendimiento de la conexión del host A en este primer escenario. La gráfica de la izquierda muestra la **tasa de transferencia por conexión** (número de bytes por segundo en el receptor) como una función de la velocidad de transmisión de la conexión. Para una velocidad de transmisión comprendida entre 0 y *R*/2, la tasa de transferencia en el receptor es igual a la velocidad de transmisión en el emisor (todo lo que envía el emisor es recibido en el receptor con un retardo finito). Sin embargo, cuando la velocidad de transmisión es mayor que *R*/2, la tasa de transferencia es de sólo *R*/2. Este límite superior de la tasa de transferencia es una consecuencia de compartir entre dos conexiones la capacidad del enlace. El enlace simplemente no puede proporcionar paquetes a un receptor a una velocidad de régimen permanente que sea mayor que *R*/2. Independientemente de lo altas que sean las velocidades de transmisión de los hosts A y B, nunca verán una tasa de transferencia mayor que *R*/2.

Alcanzar una tasa de transferencia por conexión de *R*/2 podría realmente parecer algo positivo, porque el enlace se utiliza completamente en suministrar paquetes a sus destinos. Sin embargo, la gráfica de la derecha de la Figura 3.44 muestra la consecuencia de operar cerca de la capacidad del enlace. A medida que la velocidad de transmisión se aproxima a *R*/2 (desde la izquierda), el retardo medio se hace cada vez más grande. Cuando la velocidad de transmisión excede de *R*/2, el número medio de paquetes en cola en el router no está limitado y el retardo medio entre el origen y el destino se hace infinito (suponiendo que las conexiones operan a esas velocidades de transmisión durante un periodo de tiempo infinito y que existe una cantidad infinita de espacio disponible en el buffer). Por tanto, aunque operar a

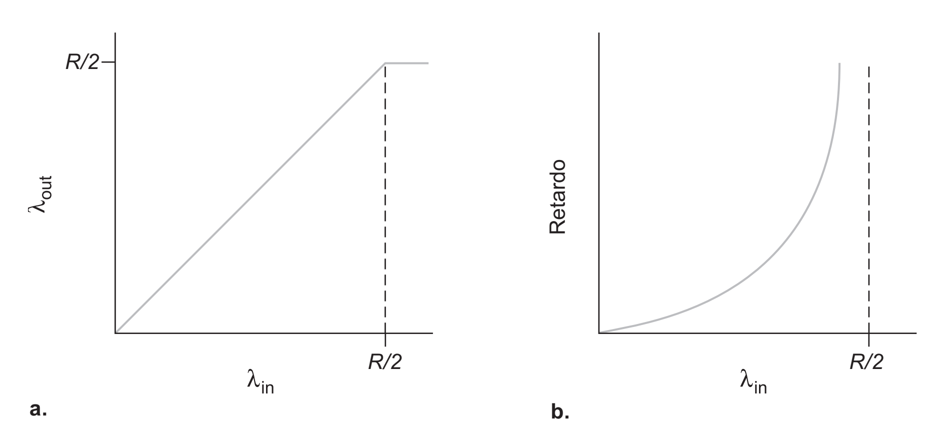


Figura 3.44 • Escenario de congestión1: tasa de transferencia y retardo en función de la velocidad de transmisión del host.

una tasa de transferencia agregada próxima a R puede ser ideal desde el punto de vista de la tasa de transferencia, no es ideal en absoluto desde el punto de vista del retardo. *Incluso en este escenario (extremadamente) idealizado, ya hemos encontrado uno de los costes de una red congestionada: los grandes retardos de cola experimentados cuando la tasa de llegada de los paquetes se aproxima a la capacidad del enlace.*

Escenario 2: dos emisores y un router con buffers finitos

Ahora vamos a modificar ligeramente el escenario 1 de dos formas (véase la Figura 3.45). En primer lugar, suponemos que el espacio disponible en los buffers del router es finito. Una consecuencia de esta suposición aplicable en la práctica es que los paquetes serán descartados cuando lleguen a un buffer que ya esté lleno. En segundo lugar, suponemos que cada conexión es fiable. Si un paquete que contiene un segmento de la capa de transporte se descarta en el router, el emisor tendrá que retransmitirlo. Dado que los paquetes pueden retransmitirse, ahora tenemos que ser más cuidadosos al utilizar el término *velocidad de transmisión.* Específicamente, vamos a designar la velocidad a la que la aplicación envía los datos originales al socket como λ_{in} bytes/segundo. La velocidad a la que la capa de transporte envía segmentos (que contienen los datos originales *y* los datos retransmitidos) a la red la denotaremos como λ'_{in} bytes/segundo. En ocasiones, λ'_{in} se denomina **carga ofrecida** a la red.

El rendimiento de este segundo escenario dependerá en gran medida de cómo se realicen las retransmisiones. En primer lugar, considere el caso no realista en el que el host A es capaz de determinar de alguna manera (mágicamente) si el buffer en el router está libre o lleno y enviar entonces un paquete sólo cuando el buffer esté libre. En este caso no se produciría ninguna pérdida, λ_{in} sería igual a λ'_{in} y la tasa de transferencia de la conexión sería igual a λ_{in}. Este caso se muestra en la Figura 3.46(a). Desde el punto de vista

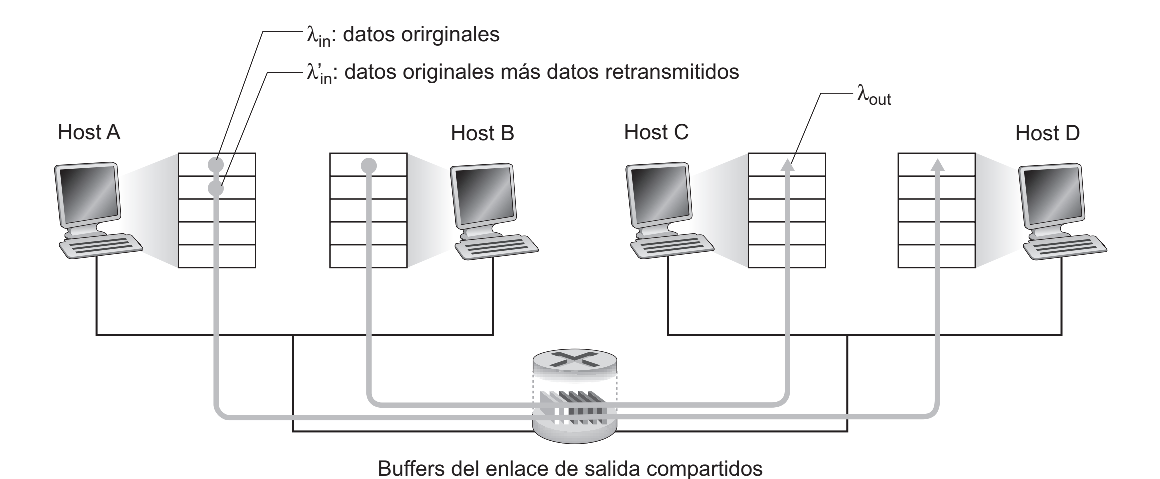


Figura 3.45 • Escenario 2: dos hosts (con retransmisión) y un router con buffers con capacidad finita.

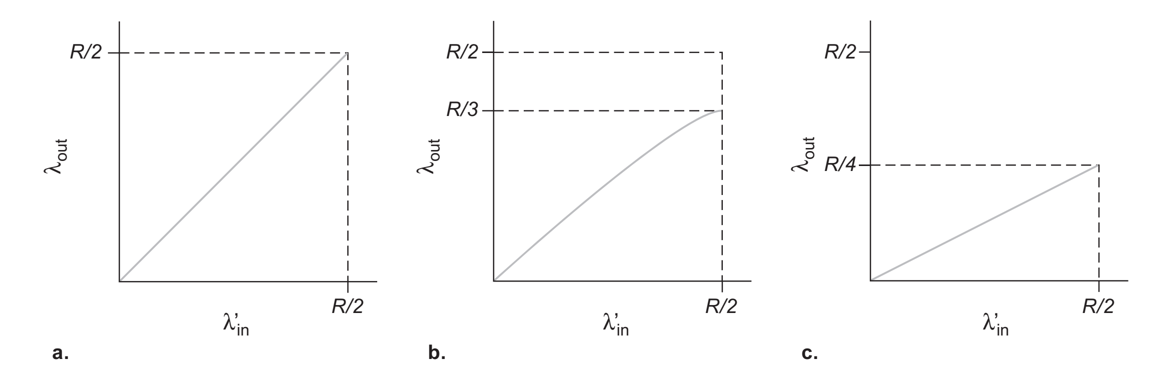


Figura 3.46 • Escenario 2: rendimiento con buffers de capacidad finita.

de la tasa de transferencia, el rendimiento es ideal: todo lo que se envía, se recibe. Observe que, en este escenario, la velocidad media de transmisión de host no puede ser mayor que *R*/2, ya que se supone que nunca tiene lugar una pérdida de paquetes.

Consideremos ahora un caso algo más realista en el que el emisor sólo retransmite cuando sabe con seguridad que un paquete se ha perdido. De nuevo, esta es una suposición un poco exagerada; sin embargo, es posible que el host emisor tenga fijado su intervalo de fin de temporización en un valor lo suficientemente grande como para garantizar que un paquete que no ha sido reconocido es un paquete que se ha perdido. En este caso, el rendimiento puede ser similar al mostrado en la Figura 3.46(b). Para apreciar lo que está ocurriendo aquí, considere el caso en el que la carga ofrecida, λ'_{in} (la velocidad de transmisión de los datos originales más la de las retransmisiones), es igual a *R*/2. Según la Figura 3.46(b), para este valor de la carga ofrecida la velocidad a la que los datos son suministrados a la aplicación del receptor es *R*/3. Por tanto, de las 0,5*R* unidades de datos transmitidos, 0,333*R* bytes/segundo (como media) son datos originales y 0,166*R* bytes/segundo (como media) son datos retransmitidos. *Tenemos aquí por tanto otro de los costes de una red congestionada: el emisor tiene que realizar retransmisiones para poder compensar los paquetes descartados (perdidos) a causa de un desbordamiento de buffer.*

Por último, considere el caso en el que el emisor puede alcanzar el fin de la temporización de forma prematura y retransmitir un paquete que ha sido retardado en la cola pero que todavía no se ha perdido. En este caso, tanto el paquete de datos original como la retransmisión pueden llegar al receptor. Por supuesto, el receptor necesitará sólo una copia de este paquete y descartará la retransmisión. En este caso, el trabajo realizado por el router al reenviar la copia retransmitida del paquete original se desperdicia, ya que el receptor ya había recibido la copia original de ese paquete. El router podría haber hecho un mejor uso de la capacidad de transmisión del enlace enviando en su lugar un paquete distinto. *Por tanto, tenemos aquí otro de los costes de una red congestionada: las retransmisiones innecesarias del emisor causadas por retardos largos pueden llevar a que un router utilice el ancho de banda del enlace para reenviar copias innecesarias de un paquete.* La Figura 3.46 (c) muestra la tasa de transferencia en función de la carga ofrecida cuando se supone que el router reenvía cada paquete dos veces (como media). Dado que cada paquete se reenvía dos veces, la tasa de transferencia tendrá un valor asintótico de *R*/4 cuando la carga ofrecida se aproxime a *R*/2.

Escenario 3: cuatro emisores, routers con buffers de capacidad finita y rutas con múltiples saltos

En este último escenario dedicado a la congestión de red tenemos cuatro hosts que transmiten paquetes a través de rutas solapadas con dos saltos, como se muestra en la Figura 3.47. De nuevo suponemos que cada host utiliza un mecanismo de fin de temporización/retransmisión para implementar un servicio de transferencia de datos fiable, que todos los hosts tienen el mismo valor de λ_{in} y que todos los enlaces de router tienen una capacidad de *R* bytes/segundo.

Consideremos la conexión del host A al host C pasando a través de los routers R1 y R2. La conexión A–C comparte el router R1 con la conexión D–B y comparte el router R2 con la conexión B–D. Para valores extremadamente pequeños de λ_{in}, es raro que el buffer se desborde (como en los dos escenarios de congestión anteriores), y la tasa de transferencia es aproximadamente igual a la carga ofrecida. Para valores ligeramente más grandes de λ_{in}, la correspondiente tasa de transferencia es también más grande, ya que se están transmitiendo más datos originales por la red y entregándose en su destino, y los desbordamientos siguen siendo raros. Por tanto, para valores pequeños de λ_{in}, un incremento de λ_{in} da lugar a un incremento de λ_{out}.

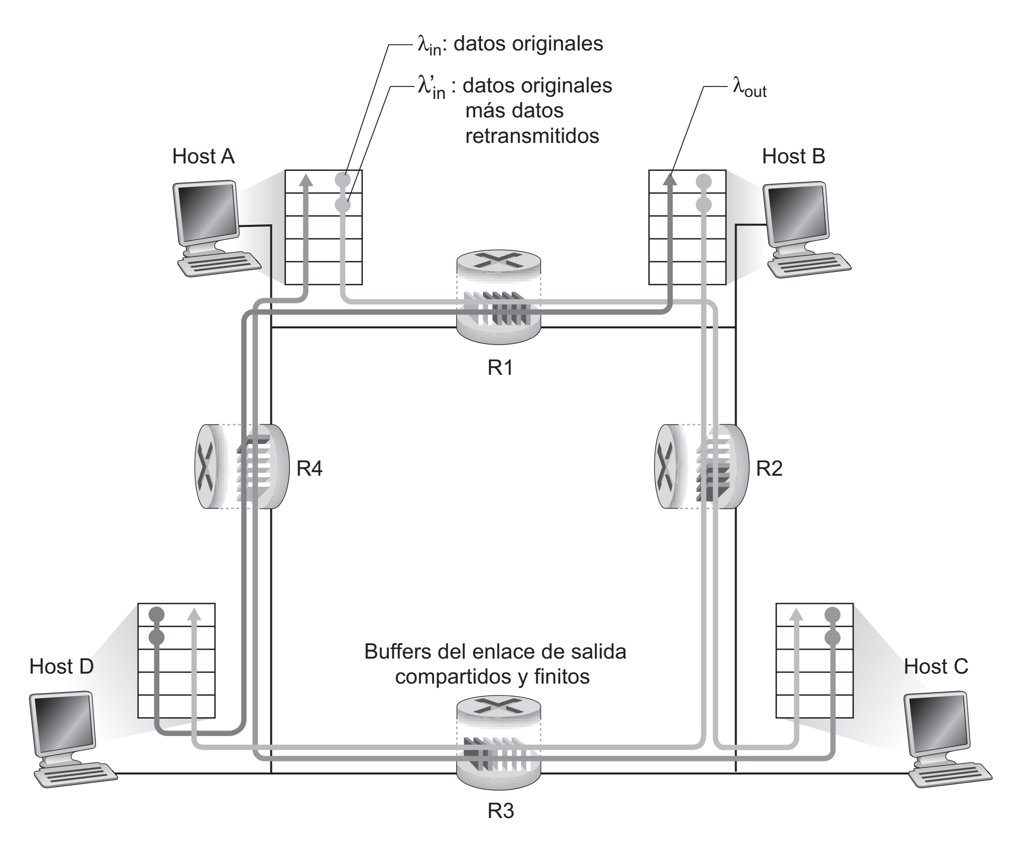


Figura 3.47 • Cuatro emisores, routers con buffers finitos y rutas con varios saltos.

Una vez considerado el caso de tráfico extremadamente bajo, pasemos a examinar el caso en que λ_{in} (y por tanto λ'_{in}) es extremadamente grande. Sea el router R2. El tráfico de A–C que llega al router R2 (el que llega a R2 después de ser reenviado desde R1) puede tener una velocidad de llegada a R2 que es como máximo *R*, la capacidad del enlace de R1 a R2, independientemente del valor de λ_{in}. Si λ'_{in} es extremadamente grande en todas las conexiones (incluyendo la conexión B–D), entonces la velocidad de llegada del tráfico de B–D a R2 puede ser mucho mayor que la del tráfico de A–C. Puesto que los tráficos de A–C y B–D tienen que competir en el router R2 por el espacio limitado disponible en el buffer, la cantidad de tráfico de A–C que consiga atravesar con éxito R2 (es decir, que no se pierda por desbordamiento del buffer) será cada vez menor a medida que la carga ofrecida por la conexión B–D aumente. En el límite, a medida que la carga ofrecida se aproxima a infinito, un buffer vacío de R2 es llenado de forma inmediata por un paquete de B–D y la tasa de transferencia de la conexión A–C en R2 tiende a cero. Esto, a su vez, *implica que la tasa de transferencia terminal a terminal de A–C tiende a cero* en el límite correspondiente a una situación de tráfico intenso. Estas consideraciones dan lugar a la relación de compromiso entre la carga ofrecida y la tasa de transferencia que se muestra en la Figura 3.48.

La razón del eventual decrecimiento de la tasa de transferencia al aumentar la carga ofrecida es evidente cuando se considera la cantidad de trabajo desperdiciado realizado por la red. En el escenario descrito anteriormente en el que había una gran cantidad de tráfico, cuando un paquete se descartaba en un router de segundo salto, el trabajo realizado por el router del primer salto al encaminar un paquete al router del segundo salto terminaba siendo "desperdiciado". Para eso, hubiera dado igual que el primer router simplemente hubiera descartado dicho paquete y hubiera permanecido inactivo, porque de todos modos el paquete no llegaría a su destino. Aún más, la capacidad de transmisión utilizada en el primer router para encaminar el paquete al segundo router podría haber sido mejor aprovechada si se hubiera empleado para transmitir un paquete diferente (por ejemplo, al seleccionar un paquete para transmitirlo, puede resultar mejor para un router dar la prioridad a los paquetes que ya hayan pasado por varios routers anteriormente). *Por tanto, aquí nos encontramos con otro de los costes de descartar un paquete a causa de la congestión de la red: cuando un paquete se descarta a lo largo de una ruta, la capacidad de transmisión empleada en cada uno de los enlaces anteriores para encaminar dicho paquete hasta el punto en el que se ha descartado termina por desperdiciarse.*

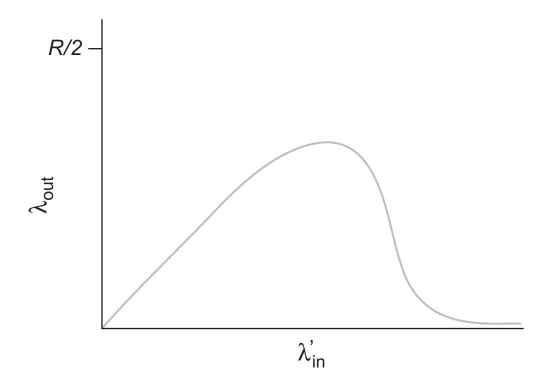


Figura 3.48 • Escenario 3: rendimiento con buffers finitos y rutas multisalto.

3.6.2 Métodos para controlar la congestión

En la Sección 3.7 examinaremos en gran detalle el método específico de TCP para controlar la congestión. Ahora, vamos a identificar los dos métodos más comunes de control de congestión que se utilizan en la práctica y abordaremos las arquitecturas de red específicas y los protocolos de control de congestión que se integran en estos métodos.

En el nivel más general, podemos diferenciar entre las técnicas de control de congestión basándonos en si la capa de red proporciona alguna ayuda explícita a la capa de transporte con propósitos de controlar la congestión:

- *Control de congestión terminal a terminal.* En este método, la capa de red *no proporciona soporte explícito* a la capa de transporte para propósitos de control de congestión. Incluso la presencia de congestión en la red tiene que ser inferida por los sistemas terminales basándose únicamente en el comportamiento observado de la red (por ejemplo, la pérdida de paquetes y los retardos). En la Sección 3.7 veremos que TCP tiene que aplicar necesariamente este método de control de congestión terminal a terminal, ya que la capa IP no proporciona ninguna realimentación a los sistemas terminales relativa a la congestión de la red. La pérdida de segmentos TCP (indicada por un fin de temporización o por un triple paquete ACK duplicado) se toma como indicación de que existe congestión en la red, por lo que TCP reduce el tamaño de su ventana en consecuencia. También veremos una propuesta más reciente para abordar el control de congestión de TCP que utiliza valores de retardo de ida y vuelta crecientes como indicadores de que existe una mayor congestión de red.
- *Control de congestión asistido por la red.* En este método de control de congestión, los componentes de la capa de red (es decir, los routers) proporcionan una realimentación explícita al emisor informando del estado de congestión en la red. Esta realimentación puede ser tan simple como un único bit que indica que existe congestión en un enlace. Este método se aplicó en las tempranas arquitecturas SNA de IBM [Schwartz 1982] y DECnet de DEC [Jain 1989; Ramakrishnan 1990], fue propuesto recientemente para las redes TCP/IP [Floyd TCP 1994; RFC 3168] y se utiliza también en el mecanismo de control de congestión de ABR (*Available bit-rate*) en las redes ATM, como hemos mencionado anteriormente. También es posible proporcionar una realimentación de red más sofisticada. Por ejemplo, una forma del mecanismo de control de congestión de ABR en las redes ATM que estudiaremos enseguida permite a un router informar explícitamente al emisor de la velocidad de transmisión que el router puede soportar en un enlace saliente. El protocolo XCP [Katabi 2002] proporciona a cada origen, en la cabecera del paquete, información de realimentación calculada por el router, indicando cómo dicho origen debe incrementar o disminuir su velocidad de transmisión.

En el mecanismo de control de congestión asistido por la red, la información acerca de la congestión suele ser realimentada de la red al emisor de una de dos formas, como se ve en la Figura 3.49. La realimentación directa puede hacerse desde un router de la red al emisor. Esta forma de notificación, normalmente, toma la forma de un **paquete de asfixia o bloqueo (*choke packet*)** (que esencialmente dice "¡Estoy congestionada!"). La segunda forma de notificación tiene lugar cuando un router marca/actualiza un campo de un paquete que se transmite del emisor al receptor para indicar que existe congestión. Después de recibir un paquete marcado, el receptor notifica al emisor la existencia de congestión. Observe que esta última forma de notificación tarda al menos un periodo igual al tiempo de ida y vuelta completo.

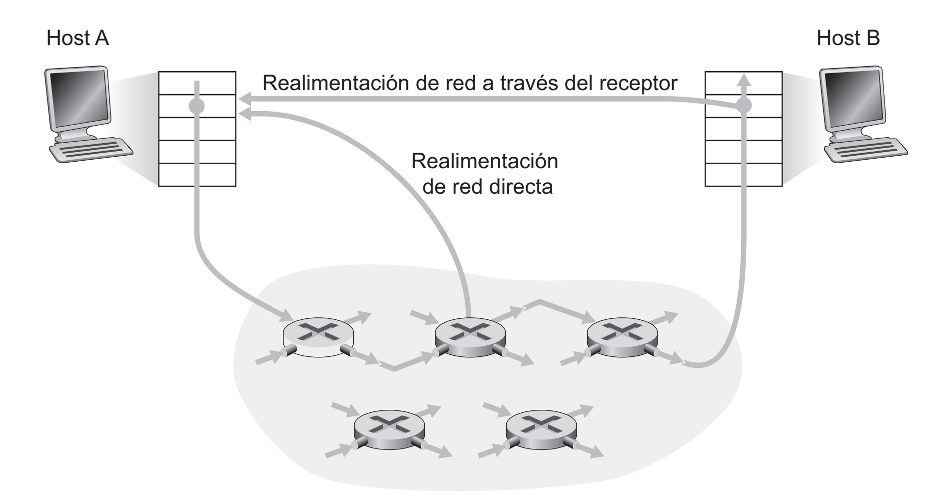


Figura 3.49 • Dos formas de realimentación de la información asistido por la red acerca de la congestión.

3.6.3 Ejemplo de control de congestion asistido por la red: control de congestión en el servicio ABR de las redes ATM

Vamos a terminar esta sección con un breve estudio del algoritmo de control de congestión del servicio ABR de las redes ATM, un protocolo que aplica un mecanismo de control de congestión asistido por la red. Tenemos que decir que nuestro objetivo en este caso no es describir en detalle los aspectos de la arquitectura ATM, sino ilustrar un protocolo que aplica un método marcadamente diferente al mecanismo de control de congestión utilizado por el protocolo TCP de Internet. De hecho, sólo abordaremos unos pocos aspectos de la arquitectura ATM que son necesarios para comprender el control de congestión del servicio ABR.

Fundamentalmente, ATM emplea para la conmutación de paquetes una técnica basada en circuitos virtuales (VC, *Virtual Circuit*). Recuerde del Capítulo 1 que esto significa que cada dispositivo de conmutación a lo largo de la ruta mantiene el estado del circuito virtual existente entre el origen y el destino. El tener información del estado de cada VC permite a un dispositivo de conmutación conocer el comportamiento de cada emisor individual (por ejemplo, conocer su velocidad media de transmisión) y llevar a cabo acciones para el control de congestión específicas para cada origen (tales como indicar explícitamente al emisor que debe reducir su velocidad cuando el conmutador comienza a congestionarse). La información del estado de cada VC conservada en los conmutadores de la red hace que las redes ATM estén idealmente adaptadas para realizar un control de congestión asistido por la red.

ABR ha sido diseñado como un servicio de transferencia de datos elástico de una forma que recuerda a TCP. Cuando la red está poco cargada, el servicio ABR debería poder aprovecharse del ancho de banda de reserva disponible; si la red está congestionada, el servicio ABR debería reducir su velocidad de transmisión a un valor mínimo predeterminado. Puede encontrar un tutorial detallado sobre la gestión de tráfico y el control de congestión del servicio ABR en redes ATM en [Jain 1996].

La Figura 3.50 muestra el marco de trabajo del mecanismo de control de congestión del servicio ABR en las redes ATM. En esta exposición, hemos adoptado la terminología de

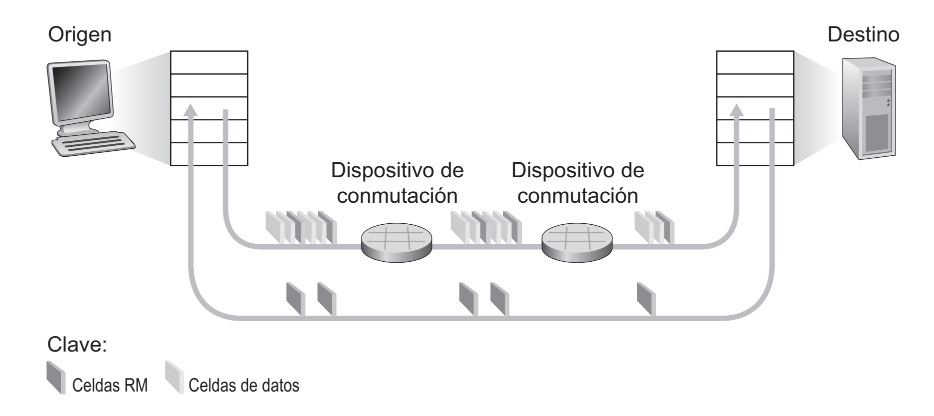


Figura 3.50 • Mecanismo de control de congestión del servicio ABR de ATM.

ATM (por ejemplo, utilizando el término *dispositivo de conmutación* en lugar de *router* y el término *celda* en lugar de *paquete*). Con el servicio ABR de ATM, las celdas de datos se transmiten desde un origen a un destino a través de una serie de dispositivos de conmutación intermedios. Intercaladas con las celdas de datos se encuentran las **celdas de gestión de recursos (celdas RM, *Resource-Management*)**; estas celdas RM se pueden utilizar para transportar información relativa al control de congestión entre los hosts y los dispositivos de conmutación. Cuando una celda RM llega a un destino, será enviada de vuelta al emisor (posiblemente después de que el destino haya modificado los contenidos de la celda RM). Un dispositivo de conmutación también puede generar una celda RM propia y enviarla directamente a un origen. Las celdas RM pueden por tanto emplearse para proporcionar tanto realimentación directa de la red como a través del receptor, como se muestra en la Figura 3.50.

El mecanismo de control de congestión del servicio ABR de ATM es un método basado en la velocidad. Es decir, el emisor calcula de forma explícita una velocidad máxima a la que puede transmitir y se autorregula de acuerdo con ello. ABR proporciona tres mecanismos para señalizar la información relativa a la congestión transmitida desde los dispositivos de conmutación al receptor:

- *Bit EFCI*. Cada *celda de datos* contiene un **bit EFCI (*Explicit Forward Congestion Indication*, Indicación de congestión explícita directa**). Un dispositivo de conmutación de la red congestionado puede poner el bit EFCI de una celda de datos a 1 para indicar que existe congestión al host de destino, el cual debe comprobar el bit EFCI de todas las celdas de datos recibidas. Cuando llega una celda RM al destino, si la celda de datos recibida más recientemente tenía el bit EFCI a 1, entonces el destino pone el bit de indicación de congestión (el bit CI) de la celda RM a 1 y devuelve la celda RM al emisor. Utilizando el bit EFCI de las celdas de datos y el bit CI de la celda RM, es posible notificar a un emisor que existe congestión en un dispositivo de conmutación de la red.
- *Bits CI y NI*. Como hemos mencionado anteriormente, las celdas RM que se transmiten del emisor al receptor están intercaladas con celdas de datos. La tasa de intercalado de las celdas RM es un parámetro ajustable, siendo el valor predeterminado una celda RM cada 32 celdas de datos. Estas celdas RM disponen de un **bit indicativo de la con-**

gestión (CI, *Congestion Indication*) y de un **bit de no incremento (NI, *No Increase*)** que un dispositivo de conmutación de la red congestionado puede configurar. Específicamente, un dispositivo de conmutación puede poner el bit NI de una celda RM que le atraviese a 1 en condiciones de congestión leve y poner a 1 el bit CI bajo condiciones de congestión severa. Cuando un host de destino recibe una celda RM, devuelve dicha celda RM al emisor con sus bits CI y NI intactos (excepto cuando el host de destino pone el bit CI a 1 como resultado del mecanismo EFCI descrito anteriormente).

- *Configuración de ER.* Cada celda RM también contiene un **campo ER (*Explicit Rate*, velocidad explícita)** de 2 bytes. Un dispositivo de conmutación congestionado puede reducir el valor contenido en el campo ER de una celda RM que le atraviese. De esta forma, el campo ER establecerá la velocidad mínima soportable de todos los dispositivos de conmutación existentes en la ruta del origen al destino.

Un emisor ABR de una red ATM ajusta la velocidad a la que puede enviar las celdas en función de los valores de CI, NI y ER contenidos en las celdas RM devueltas. Las reglas para llevar a cabo este ajuste de velocidad son bastante complejas y algo tediosas. El lector interesado en este tema puede consultar [Jain 1996] para conocer los detalles.



3.7 Mecanismo de control de congestión de TCP

En esta sección vamos a continuar con nuestro estudio de TCP. Como hemos visto en la Sección 3.5, TCP proporciona un servicio de transporte fiable entre dos procesos que se ejecutan en hosts diferentes. Otro componente clave de TCP es su mecanismo de control de congestión. Como hemos mencionado en la sección anterior, TCP tiene que utilizar un control de congestión terminal a terminal en lugar de un control de congestión asistido por la red, ya que la capa IP no proporciona una realimentación explícita a los sistemas terminales en lo tocante a la congestión de la red.

El método empleado por TCP consiste en que cada emisor limite la velocidad a la que transmite el tráfico a través de su conexión en función de la congestión de red percibida. Si un emisor TCP percibe que en la ruta entre él mismo y el destino apenas existe congestión, entonces incrementará su velocidad de transmisión; por el contario, si el emisor percibe que existe congestión a lo largo de la ruta, entonces reducirá su velocidad de transmisión. Pero este método plantea tres cuestiones. En primer lugar, ¿cómo limita el emisor TCP la velocidad a la que envía el tráfico a través de su conexión? En segundo lugar, ¿cómo percibe el emisor TCP que existe congestión en la ruta entre él mismo y el destino? Y, tercero, ¿qué algoritmo deberá emplear el emisor para variar su velocidad de transmisión en función de la congestión percibida terminal a terminal?

Examinemos en primer lugar cómo un emisor TCP limita la velocidad a la que envía el tráfico a través de su conexión. En la Sección 3.5, hemos visto que cada lado de una conexión TCP consta de un buffer de recepción, un buffer de transmisión y varias variables (`UltimoByteLeido`, `VentanaRecepcion`, etc.). El mecanismo de control de congestión de TCP que opera en el emisor hace un seguimiento de una variable adicional, la **ventana de congestión**. Esta ventana, indicada como `VentanaCongestion`, impone una restricción sobre la velocidad a la que el emisor TCP puede enviar tráfico a la red. Específicamente, la cantidad de datos no reconocidos en un emisor no puede exceder el mínimo de entre `VentanaCongestion` y `VentanaRecepcion`, es decir:

```
UltimoByteLeido – UltimoByteReconocido ≤
                        mín{VentanaCongestión, VentanaRecepcion}
```

Con el fin de centrarnos en el mecanismo de control de congestión (en oposición al control de flujo), vamos a suponer que el buffer de recepción TCP es tan grande que la restricción de la ventana de recepción puede ignorarse; por tanto, la cantidad de datos no reconocidos por el emisor queda limitada únicamente por `VentanaCongestion`. Supondremos también que el emisor siempre tiene datos que enviar; es decir, todos los segmentos de la ventana de congestión son enviados.

La restricción anterior limita la cantidad de datos no reconocidos por el emisor y, por tanto, limita de manera indirecta la velocidad de transmisión del emisor. Para comprender esto imagine una conexión en la que tanto la pérdida de paquetes como los retardos de transmisión sean despreciables. En esta situación, lo que ocurre a grandes rasgos es lo siguiente: al inicio de cada periodo RTT, la restricción permite al emisor enviar `ventanacongestion` bytes de datos a través de la conexión; al final del periodo RTT, el emisor recibe los paquetes ACK correspondientes a los datos. *Por tanto, la velocidad de transmisión del emisor es aproximadamente igual a VentanaCongestion/RTT bytes/segundo. Ajustando el valor de la ventana de congestión, el emisor puede ajustar la velocidad a la que transmite los datos a través de su conexión.*

Veamos ahora cómo percibe un emisor TCP que existe congestión en la ruta entre él mismo y el destino. Definamos un "suceso de pérdida" en un emisor TCP como el hecho de que se produzca un fin de temporización o se reciban tres paquetes ACK duplicados procedentes del receptor (recuerde la exposición de la Sección 3.5.4 sobre el suceso de fin de temporización mostrado en la Figura 3.33 y la subsiguiente modificación para incluir la retransmisión rápida a causa de la recepción de tres paquetes ACK duplicados). Cuando existe una congestión severa, entonces uno o más de los buffers de los routers existentes a lo largo de la ruta pueden desbordarse, dando lugar a que un datagrama (que contenga un segmento TCP) sea descartado. A su vez, el datagrama descartado da lugar a un suceso de pérdida en el emisor (debido a un fin de temporización o a la recepción de tres paquetes ACK duplicados), el cual lo interpreta como una indicación de que existe congestión en la ruta entre el emisor y el receptor.

Ahora que ya hemos visto cómo se detecta la existencia de congestión en la red, vamos a considerar el mejor de los casos, cuando no existe congestión en la red, es decir, cuando no se producen pérdidas de paquetes. En este caso, el emisor TCP recibirá los paquetes de reconocimiento ACK correspondientes a los segmentos anteriormente no reconocidos. Como veremos, TCP interpretará la llegada de estos paquetes ACK como una indicación de que todo está bien (es decir, que los segmentos que están siendo transmitidos a través de la red están siendo entregados correctamente al destino) y empleará esos paquetes de reconocimiento para incrementar el tamaño de la ventana de congestión (y, por tanto, la velocidad de transmisión). Observe que si la velocidad de llegada de los paquetes ACK es lenta, porque por ejemplo la ruta terminal a terminal presenta un retardo grande o contiene un enlace con un ancho de banda pequeño, entonces el tamaño de la ventana de congestión se incrementará a una velocidad relativamente lenta. Por el contrario, si la velocidad de llegada de los paquetes de reconocimiento es alta, entonces el tamaño de la ventana de congestión se incrementará más rápidamente. Puesto que TCP utiliza los paquetes de reconocimiento para provocar (o temporizar) sus incrementos del tamaño de la ventana de congestión, se dice que TCP es **auto-temporizado**.

Conocido el *mecanismo* de ajuste del valor de `VentanaCongestion` para controlar la velocidad de transmisión, la cuestión crítica que nos queda es: *¿cómo* debería un emisor TCP determinar su velocidad de transmisión? Si los emisores TCP de forma colectiva transmiten a velocidades demasiado altas pueden congestionar la red, llevándola al tipo de colapso de congestión que hemos visto en la Figura 3.48. De hecho, la versión de TCP que vamos a estudiar a continuación fue desarrollada en respuesta al colapso de congestión observado en Internet [Jacobson 1988] en las versiones anteriores de TCP. Sin embargo, si los emisores TCP son demasiado cautelosos y transmiten la información muy lentamente, podrían infrautilizar el ancho de banda de la red; es decir, los emisores TCP podrían transmitir a velocidades más altas sin llegar a congestionar la red. Entonces, ¿cómo pueden determinar los emisores TCP sus velocidades de transmisión de manera que no congestionen la red a la vez que hacen uso del todo el ancho de banda disponible? ¿Están los emisores TCP explícitamente coordinados, o existe un método distribuido en el que dichos emisores TCP puedan establecer sus velocidades de transmisión basándose únicamente en la información local? TCP responde a estas preguntas basándose en los siguientes principios:

- *Un segmento perdido implica congestión y, por tanto, la velocidad del emisor TCP debe reducirse cuando se pierde un segmento*. Recuerde que en la Sección 3.5.4 hemos visto que un suceso de fin de temporización o la recepción de cuatro paquetes de reconocimiento para un segmento dado (el paquete ACK original y los tres duplicados) se interpreta como una indicación implícita de que se ha producido un "suceso de pérdida" del segmento que sigue al segmento que ha sido reconocido cuatro veces, activando el proceso de retransmisión del segmento perdido. Desde el punto de vista del mecanismo de control de congestión, la cuestión es cómo el emisor TCP debe disminuir el tamaño de su ventana de congestión y, por tanto, su velocidad de transmisión, en respuesta a este suceso de pérdida inferido.
- *Un segmento que ha sido reconocido indica que la red está entregando los segmentos del emisor al receptor y, por tanto, la velocidad de transmisión del emisor puede incrementarse cuando llega un paquete ACK correspondiente a un segmento que todavía no había sido reconocido*. La llegada de paquetes de reconocimiento se interpreta como una indicación implícita de que todo funciona bien (los segmentos están siendo entregados correctamente del emisor al receptor y la red por tanto no está congestionada). Luego se puede aumentar el tamaño de la ventana de congestión.
- *Tanteo del ancho de banda*. Puesto que los paquetes ACK indican que la ruta entre el origen y el destino está libre de congestión y la pérdida de paquetes indica que hay una ruta congestionada, la estrategia de TCP para ajustar su velocidad de transmisión consiste en incrementar su velocidad en respuesta a la llegada de paquetes ACK hasta que se produce una pérdida, momento en el que reduce la velocidad de transmisión. El emisor TCP incrementa entonces su velocidad de transmisión para tantear la velocidad a la que de nuevo aparece congestión, retrocede a partir de ese punto y comienza de nuevo a tantear para ver si ha variado la velocidad a la que comienza de nuevo a producirse congestión. El comportamiento del emisor TCP es quizá similar a la del niño que pide (y consigue) una y otra vez golosinas hasta que se le dice "¡No!", momento en el que da marcha atrás, pero enseguida comienza otra vez a pedir más golosinas. Observe que la red no proporciona una indicación explícita del estado de congestión (los paquetes ACK y las pérdidas se utilizan como indicadores implícitos) y que cada emisor TCP reacciona a la información local de forma asíncrona con respecto a otros emisores TCP.

Ahora que ya conocemos los fundamentos del mecanismo de control de congestión de TCP, estamos en condiciones de pasar a estudiar los detalles del famoso **algoritmo de control de congestión de TCP**, que fue descrito por primera vez en el libro de [Jacobson 1988] y que está estandarizado en el documento [RFC 2581]. El algoritmo consta de tres componentes principales: (1) arranque lento (*slow start*), (2) evitación de la congestión (*congestion avoidance*) y (3) recuperación rápida (*fast recovery*). Los dos primeros componentes son obligatorios en TCP, diferenciándose en la forma en que aumentan el tamaño de la ventana de congestión en respuesta a los paquetes ACK recibidos. Vamos a ver brevemente que el arranque lento incrementa el tamaño de la ventana de congestión más rápidamente (¡contrariamente a lo que indica su nombre!) que la evitación de la congestión. El componente recuperación rápida es recomendable, aunque no obligatorio, para los emisores TCP.

Arranque lento

Cuando se inicia una conexión TCP, el valor de la ventana de congestión (`VentanaCongestion`) normalmente se inicializa con un valor pequeño igual a 1 MSS (tamaño máximo de segmento) [RFC 3390], que da como resultado una velocidad de transmisión inicial aproximadamente igual a MSS/RTT. Por ejemplo, si MSS = 500 bytes y RTT = 200 milisegundos, la velocidad de transmisión inicial será aproximadamente de 20 kbps. Puesto que el ancho de banda disponible para el emisor TCP puede ser mucho más grande que el valor de MSS/ RTT, al emisor TCP le gustaría poder determinar rápidamente la cantidad de ancho de banda disponible. Por tanto, en el estado de **arranque lento**, el valor de `VentanaCon-`

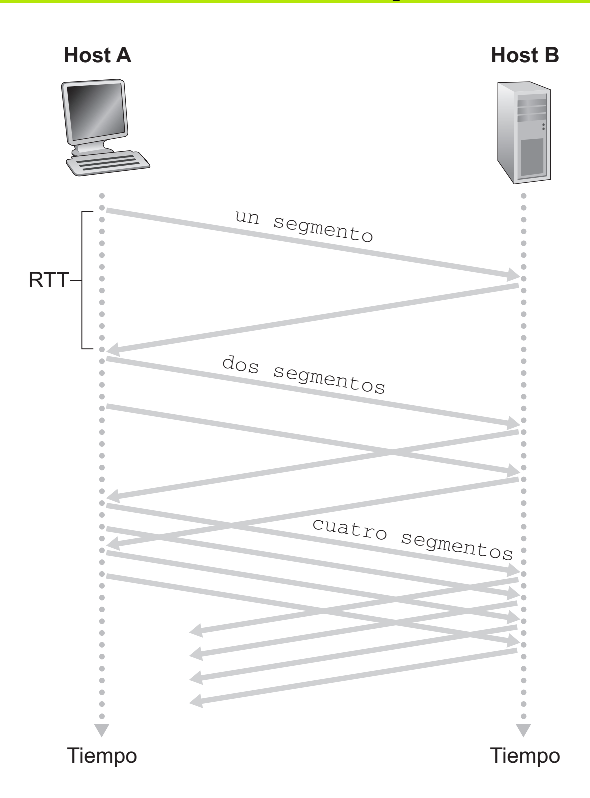


Figura 3.51 • Fase de arranque lento de TCP.

`gestion` se establece en 1 MSS y se incrementa 1 MSS cada vez que se produce el primer reconocimiento de un segmento transmitido. En el ejemplo de la Figura 3.51, TCP envía el primer segmento a la red y espera el correspondiente paquete ACK. Cuando llega dicho paquete de reconocimiento, el emisor TCP incrementa el tamaño de la ventana de congestión en 1 MSS y transmite dos segmentos de tamaño máximo. Estos segmentos son entonces reconocidos y el emisor incrementa el tamaño de la ventana de congestión en 1 MSS por cada uno de los segmentos de reconocimiento, generando así una ventana de congestión de 4 MSS, etc. Este proceso hace que la velocidad de transmisión se duplique en cada periodo RTT. Por tanto, la velocidad de transmisión inicial de TCP es baja, pero crece exponencialmente durante esa fase de arranque lento.

Pero, ¿cuándo debe finalizar este crecimiento exponencial? El algoritmo del arranque lento proporciona varias respuestas a esta cuestión. En primer lugar, si se produce un suceso de pérdida de paquete (es decir, si hay congestión) señalado por un fin de temporización, el emisor TCP establece el valor de `VentanaCongestion` en 1 e inicia de nuevo un proceso de arranque lento. También define el valor de una segunda variable de estado que establece el umbral del arranque lento y que denominaremos `umbral` en `VentanaCongestion/2`, la mitad del valor del tamaño de la ventana de congestión cuando se ha detectado que existe congestión. La segunda forma en la que la fase de arranque lento puede terminar está directamente ligada al valor de `umbral`. Dado que `umbral` es igual a la mitad del valor que `VentanaCongestion` tenía cuando se detectó congestión por última vez, puede resultar algo imprudente continuar duplicando el valor de `VentanaCongestion` cuando se alcanza o sobrepasa el valor de `umbral`. Por tanto, cuando el valor de `VentanaCongestion` es igual a `umbral`, la fase de arranque lento termina y las transacciones TCP pasan al modo de evitación de la congestión. Como veremos, TCP incrementa con más cautela el valor de `VentanaCongestion` cuando está en el modo de evitación de la congestión. La última forma en la que puede terminar la fase de arranque lento es si se detectan tres paquetes ACK duplicados, en cuyo caso TCP realiza una retransmisión rápida (véase la Sección 3.5.4) y entra en el estado de recuperación rápida, que veremos más adelante. El comportamiento de TCP en la fase de arranque lento se resume en la descripción de la FSM del control de congestión de TCP de la Figura 3.52. El algoritmo de arranque lento tiene sus raíces en [Jacobson 1988]; en [Jain 1986] se proponía, de forma independiente, un método similar al algoritmo de arranque lento.

Evitación de la congestión

Al entrar en el estado de evitación de la congestión, el valor de `VentanaCongestion` es aproximadamente igual a la mitad de su valor en el momento en que se detectó congestión por última vez (podemos estar, por tanto, al borde de la congestión). En consecuencia, en lugar de duplicar el valor de `VentanaCongestion` para cada RTT, TCP adopta un método más conservador e incrementa el valor de `VentanaCongestion` solamente en un MSS cada RTT [RFC 2581]. Esto puede llevarse a cabo de varias maneras. Un método habitual consiste en que un emisor TCP aumenta el valor de `VentanaCongestion` en MSS bytes (MSS/`VentanaCongestion`) cuando llega un nuevo paquete de reconocimiento. Por ejemplo, si MSS es igual a 1.460 bytes y `VentanaCongestion` es igual a 14.600 bytes, entonces se enviarán 10 segmentos en un RTT. Cada ACK que llega (suponiendo un ACK por segmento) incrementa el tamaño de la ventana de congestión en 1/10 MSS y, por tanto, el valor del tamaño de la ventana de congestión habrá aumentado en un MSS después de los ACK correspondientes a los 10 segmentos que hayan sido recibidos.

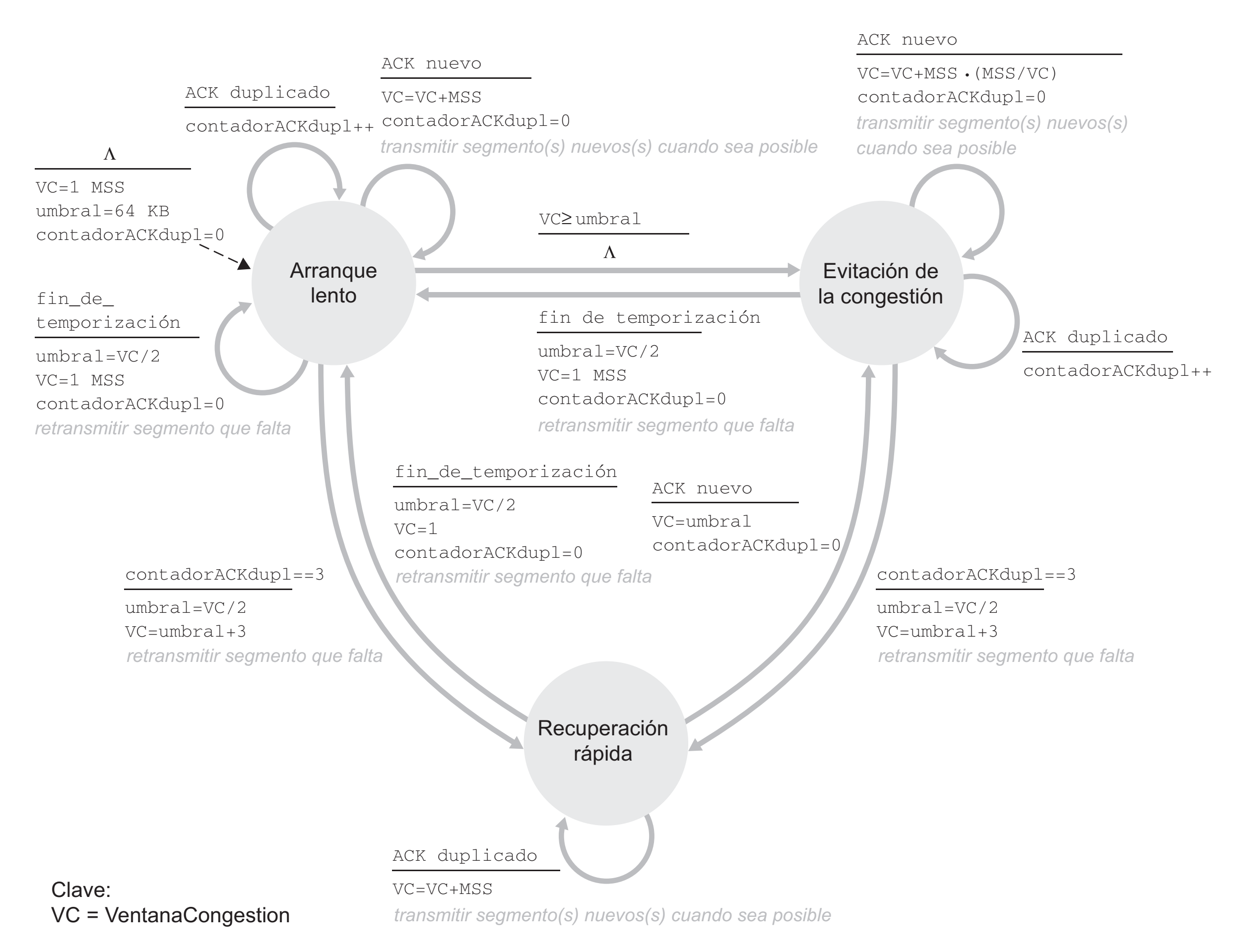


Figura 3.52 • Descripción de la máquina de estados finitos del mecanismo de control de congestión de TCP.

Pero, ¿en qué momento debería detenerse el crecimiento lineal (1 MSS por RTT) en el modo de evitación de la congestión? El algoritmo de evitación de la congestión de TCP se comporta del mismo modo que cuando tiene lugar un fin de temporización. Como en el caso del modo de arranque lento, el valor de `VentanaCongestion` se fija en 1 MSS y el valor de `umbral` se actualiza haciéndose igual a la mitad del valor de `VentanaCongestion` cuando se produce un suceso de pérdida de paquete. Sin embargo, recuerde que también puede detectarse una pérdida de paquete a causa de la llegada de tres ACK duplicados. En este caso, la red continúa entregando segmentos del emisor al receptor (como señala la recepción de paquetes ACK duplicados). Por tanto, el comportamiento de TCP ante este tipo de pérdida debería ser menos drástico que ante una pérdida de paquete indicada por un fin de temporización: TCP divide entre dos el valor de `VentanaCongestion` (añadiendo 3 MSS como forma de tener en cuenta los tres ACK duplicados que se han recibido) y configura el valor de `umbral` para que sea igual a la mitad del valor que `VentanaCongestion` tenía cuando se recibieron los tres ACK duplicados. A continuación, se entra en el estado de recuperación rápida.

Recuperación rápida

En la fase de recuperación rápida, el valor de `VentanaCongestion` se incrementa en 1 MSS por cada ACK duplicado recibido correspondiente al segmento que falta y que ha causado que TCP entre en el estado de recuperación rápida. Cuando llega un ACK para el segmento que falta, TCP entra de nuevo en el estado de evitación de la congestión después de disminuir el valor de `VentanaCongestion`. Si se produce un fin de temporización, el mecanismo de recuperación rápida efectúa una transición al estado de arranque lento después de realizar las mismas acciones que en los modos de arranque lento y de evitación de la congestión: el valor de `VentanaCongestion` se establece en 1 MSS y el valor de `umbral` se hace igual a la mitad del valor que tenía `VentanaCongestion` cuando tuvo lugar el suceso de pérdida.

El mecanismo de recuperación rápida es un componente de TCP recomendado, aunque no obligatorio [RFC 2581]. Es interesante resaltar que una versión anterior de TCP, conocida como **TCP Tahoe**, establece incondicionalmente el tamaño de la ventana de congestión en 1 MSS y entra en el estado de arranque lento después de un suceso de pérdida indicado por un fin de temporización o por la recepción de tres ACK duplicados. La versión más reciente de TCP, **TCP Reno**, incorpora la recuperación rápida.

La Figura 3.53 ilustra la evolución de la ventana de congestión de TCP para Reno y Tahoe. En esta figura, inicialmente el umbral es igual a 8 MSS. Durante los ocho primeros ciclos de transmisión, Tahoe y Reno realizan acciones idénticas. La ventana de congestión crece rápidamente de forma exponencial durante la fase de arranque lento y alcanza el umbral en el cuarto ciclo de transmisión. A continuación, la ventana de congestión crece linealmente hasta que se produce un suceso de tres ACK duplicados, justo después del octavo ciclo de transmisión. Observe que el tamaño de la ventana de congestión es igual a 12 • MSS cuando se produce el suceso de pérdida. El valor de `umbral` se hace entonces igual a 0,5 • `VentanaCongestion` = 6 • MSS. En TCP Reno, el tamaño de la ventana de congestión es puesto a `VentanaCongestion` = 6 • MSS y luego crece linealmente. En TCP Tahoe, la ventana de congestión es igual a 1 MSS y crece exponencialmente hasta que alcanza el valor de `umbral`, punto a partir del cual crece linealmente.

La Figura 3.52 presenta la descripción completa de la máquina de estados finitos de los algoritmos del mecanismo de control de congestión de TCP: arranque lento, evitación de la congestión y recuperación rápida. En la figura también se indica dónde pueden producirse

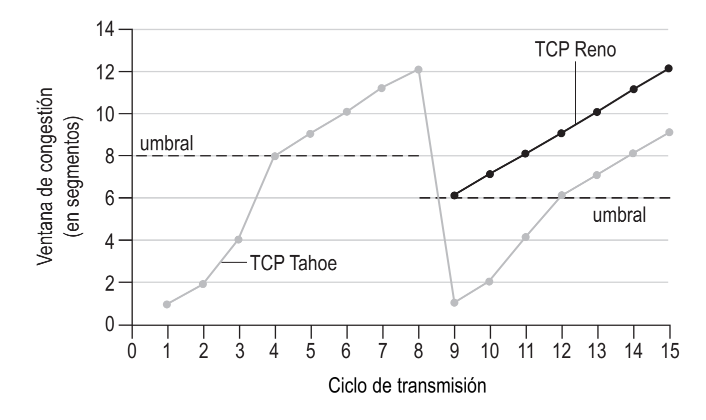


Figura 3.53 • Evolución de la ventana de congestión de TCP (Tahoe y Reno).

transmisiones de nuevos segmentos y dónde retransmisiones de segmentos. Aunque es importante diferenciar entre las retransmisiones/control de errores de TCP y el control de congestión de TCP, también lo es apreciar cómo estos dos aspectos de TCP están estrechamente vinculados.

Control de congestión de TCP: retrospectiva

Una vez vistos los detalles de las fases de arranque lento, de evitación de la congestión y de recuperación rápida, merece la pena retroceder un poco para clarificar las cosas. Ignorando la fase inicial de arranque lento en la que se establece la conexión y suponiendo que las pérdidas están indicadas por la recepción de tres ACK duplicados en lugar de por fines de temporización, el control de congestión de TCP consiste en un crecimiento lineal (aditivo) de `VentanaCongestion` a razón de 1 MSS por RTT, seguido de un decrecimiento multiplicativo (división entre dos) del tamaño de la ventana, `VentanaCongestion`, cuando se reciben tres ACK duplicados. Por esta razón, suele decirse que el control de congestión de TCP es una forma de **crecimiento aditivo y decrecimiento multiplicativo (AIMD, *Additive-Increase, Multiplicative-Decrease*)** de control de congestión. El control de congestión AIMD presenta un comportamiento en forma de "diente de sierra", como se muestra en la Figura 3.54, lo que también ilustra nuestra anterior intuición de que TCP "va tanteando" el ancho de banda. (TCP aumenta linealmente el tamaño de su ventana de congestión, y por tanto su velocidad de transmisión, hasta que tiene lugar la recepción de tres ACK duplicados. A continuación, divide entre dos su ventana de congestión, pero vuelve después a crecer linealmente, tanteando para ver si hay disponible ancho de banda adicional.)

Como hemos mencionado anteriormente, la mayor parte de las implementaciones TCP actuales emplean el algoritmo Reno [Padhye 2001]. Se han propuesto muchas variantes del algoritmo Reno [RFC 3782; RFC 2018]. El algoritmo TCP Vegas [Brakmo 1995; Ahn 1995] intenta evitar la congestión manteniendo una buena tasa de transferencia. La idea básica del algoritmo Vegas es (1) detectar la congestión en los routers existentes entre el origen y el destino *antes* de que se pierda un paquete y (2) reducir la velocidad linealmente cuando se detecta una pérdida inminente de paquetes. La pérdida inminente de un paquete se predice observando el RTT. Cuanto mayor es el RTT de los paquetes, mayor es la congestión en los

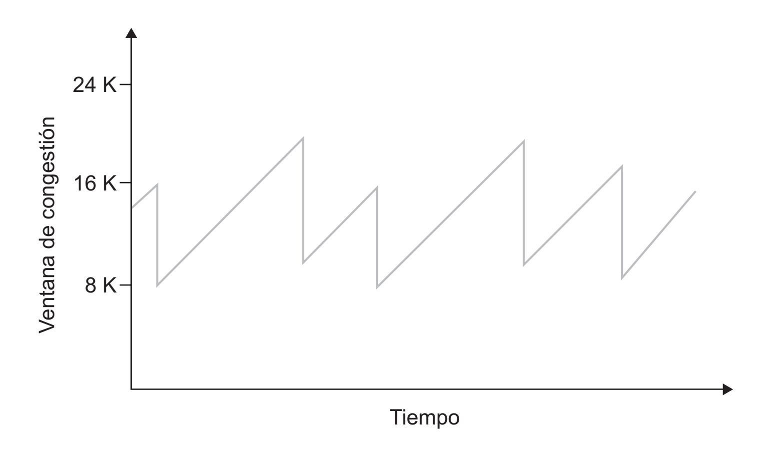


Figura 3.54 • Control de congestión con crecimiento aditivo y decrecimiento multiplicativo.

routers. Linux soporta una serie de algoritmos de control de congestión (incluyendo TCP Reno y TCP Vegas) y permite a un administrador del sistema configurar la versión de TCP que se va a utilizar. La versión predeterminada de TCP en la versión de Linux 2.6.18 es CUBIC [Ha 2008], una versión de TCP desarrollada para aplicaciones de gran ancho de banda.

El algoritmo AIMD de TCP fue desarrollado basándose en un enorme trabajo de ingeniería y de experimentación con los mecanismos de control de congestión en redes reales. Diez años después del desarrollo de TCP, los análisis teóricos mostraron que el algoritmo de control de congestión de TCP sirve como un algoritmo de optimización asíncrona distribuido que da como resultado la optimización simultánea de diversos aspectos importantes, tanto de las prestaciones proporcionadas al usuario como del rendimiento de la red [Kelly 1998]. Desde entonces se han desarrollado muchos aspectos teóricos del control de congestión [Srikant 2004].

Descripción macroscópica de la tasa de transferencia de TCP

Visto el comportamiento en diente de sierra de TCP, resulta natural preguntarse cuál es la tasa de transferencia media (es decir, la velocidad media) de una conexión TCP de larga duración. En este análisis vamos a ignorar las fases de arranque lento que tienen lugar después de producirse un fin de temporización (estas fases normalmente son muy cortas, ya que el emisor sale de ellas rápidamente de forma exponencial). Durante un intervalo concreto de ida y vuelta, la velocidad a la que TCP envía datos es función del tamaño de la ventana de congestión y del *RTT* actual. Cuando el tamaño de la ventana es *w* bytes y el tiempo actual de ida y vuelta es *RTT* segundos, entonces la velocidad de transmisión de TCP es aproximadamente igual a *w/RTT*. TCP comprueba entonces si hay ancho de banda adicional incrementando *w* en 1 MSS cada *RTT* hasta que se produce una pérdida. Sea *W* el valor de *w* cuando se produce una pérdida. Suponiendo que *RTT* y *W* son aproximadamente constantes mientras dura la conexión, la velocidad de transmisión de TCP varía entre $W/(2 \cdot RTT)$ y *W/RTT*.

Estas suposiciones nos llevan a un modelo macroscópico extremadamente simplificado del comportamiento en régimen permanente de TCP. La red descarta un paquete de la conexión cuando la velocidad aumenta hasta *W/RTT;* la velocidad entonces se reduce a la mitad y luego aumenta MSS/*RTT* cada *RTT* hasta que de nuevo alcanza el valor *W/RTT*. Este proceso se repite una y otra vez. Puesto que la tasa de transferencia (es decir, la velocidad) de TCP aumenta linealmente entre los dos valores extremos, tenemos

$$\text{tasa de transferencia media de una conexión} = \frac{0{,}75 \cdot W}{RTT}$$

Utilizando este modelo altamente idealizado para la dinámica del régimen permanente de TCP, también podemos deducir una expresión interesante que relaciona la tasa de pérdidas de una conexión con su ancho de banda disponible [Mahdavi 1997]. Dejamos esta demostración para los problemas de repaso. Un modelo más sofisticado que, según se ha comprobado, concuerda empíricamente con los datos medidos es el que se describe en [Padhye 2000].

El futuro de TCP

Es importante darse cuenta de que el control de congestión de TCP ha ido evolucionando a lo largo de los años y todavía sigue evolucionando. Puede leer un resumen sobre el meca-

nismo del control de congestión de TCP realizado a finales de la década de 1990 en [RFC 2581]; si desea ver una exposición acerca de otros desarrollos sobre el control de congestión de TCP, consulte [Floyd 2001]. Lo que era bueno para Internet cuando la mayoría de las conexiones TCP transportaban tráfico SMTP, FTP y Telnet no es necesariamente bueno para la Internet actual, en la que domina HTTP, o para una futura Internet que proporcione servicios que hoy ni siquiera podemos imaginar.

La necesidad de una evolución continua de TCP puede ilustrarse considerando las conexiones TCP de alta velocidad necesarias para las aplicaciones de computación reticular (*grid-computing*) [Foster 2002]. Por ejemplo, considere una conexión TCP con segmentos de 1.500 bytes y un *RTT* de 100 milisegundos y suponga que deseamos enviar datos a través de esta conexión a 10 Gbps. Siguiendo el documento [RFC 3649], observamos que utilizar la fórmula anterior para la tasa de transferencia de TCP, con el fin de alcanzar una tasa de transferencia de 10 Gbps, nos daría un tamaño medio de la ventana de congestión de 83.333 segmentos, que son *muchos* segmentos, lo que despierta el temor a que uno de esos 83.333 segmentos en tránsito pueda perderse. ¿Qué ocurriría si se produjeran pérdidas? O, dicho de otra manera, ¿qué fracción de los segmentos transmitidos podría perderse que permitiera al algoritmo de control de congestión de TCP de la Figura 3.52 alcanzar la velocidad deseada de 10 Gbps? En las cuestiones de repaso de este capítulo, mostraremos cómo derivar una fórmula que expresa la tasa de transferencia de una conexión TCP en función de la tasa de pérdidas (*L*), el tiempo de ida y vuelta (*RTT*) y el tamaño máximo de segmento (MSS):

$$\text{tasa de transferencia media de una conexión} = \frac{1{,}22 \ \ MSS}{RTT\sqrt{L}}$$

Con esta fórmula, podemos ver que para alcanzar una tasa de transferencia de 10 Gbps, el actual algoritmo de control de congestión de TCP sólo puede tolerar una probabilidad de pérdida de segmentos de $2 \cdot 10^{-10}$ (o, lo que es equivalente, un suceso de pérdida por cada 5.000.000.000 segmentos), lo cual es una tasa muy baja. Esta observación ha llevado a una serie de investigadores a buscar nuevas versiones de TCP que estén diseñadas específicamente para estos entornos de alta velocidad; consulte [Jin 2004; RFC 3649; Kelly 2003; Ha 2008] para obtener información sobre estos trabajos.

3.7.1 Equidad

Considere ahora *K* conexiones TCP, cada una de ellas con una ruta terminal a terminal diferente, pero atravesando todas ellas un enlace de cuello de botella con una velocidad de transmisión de *R* bps (con *enlace de cuello de botella* queremos decir que, para cada conexión, todos los restantes enlaces existentes a lo largo de la ruta de la conexión no están congestionados y tienen una capacidad de transmisión grande comparada con la capacidad de transmisión del enlace de cuello de botella). Suponga que cada conexión está transfiriendo un archivo de gran tamaño y que no existe tráfico UDP atravesando el enlace de cuello de botella. Se dice que un mecanismo de control de congestión es *equitativo* si la velocidad media de transmisión de cada conexión es aproximadamente igual a *R/K;* es decir, cada conexión obtiene la misma cuota del ancho de banda del enlace.

¿Es el algoritmo AIMD de TCP equitativo, teniendo en cuenta que diferentes conexiones TCP pueden iniciarse en instantes distintos y, por tanto, pueden tener distintos tamaños de ventana en un instante determinado? [Chiu 1989] proporciona una explicación elegante e intuitiva de por qué el control de congestión de TCP converge para proporcionar la misma

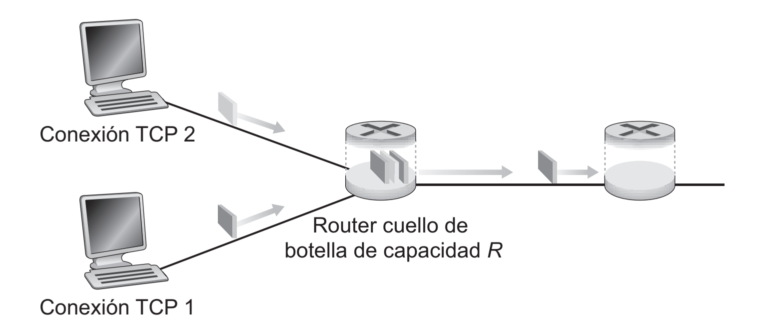


Figura 3.55 • Dos conexiones TCP que comparten un mismo enlace de cuello de botella.

cuota de ancho de banda de un enlace de cuello de botella a las conexiones TCP que compiten por el ancho de banda.

Consideremos el caso simple de dos conexiones TCP que comparten un mismo enlace, cuya velocidad de transmisión es R, como se muestra en la Figura 3.55. Suponemos que las dos conexiones tienen el mismo MSS y el mismo RTT (por lo que si tienen el mismo tamaño de ventana de congestión, entonces tienen la misma tasa de transferencia). Además, tienen que enviar una gran cantidad de datos y ninguna otra conexión TCP ni datagrama UDP atraviesan este enlace compartido. Asimismo, vamos a ignorar la fase de arranque lento de TCP y vamos a suponer que las conexiones TCP están operando en el modo de evitación de la congestión (AIMD) durante todo el tiempo.

La gráfica de la Figura 3.56 muestra la tasa de transferencia de las dos conexiones TCP. Si TCP está diseñado para que las dos conexiones compartan equitativamente el ancho de banda del enlace, entonces la tasa de transferencia alcanzada debería caer a lo largo de la

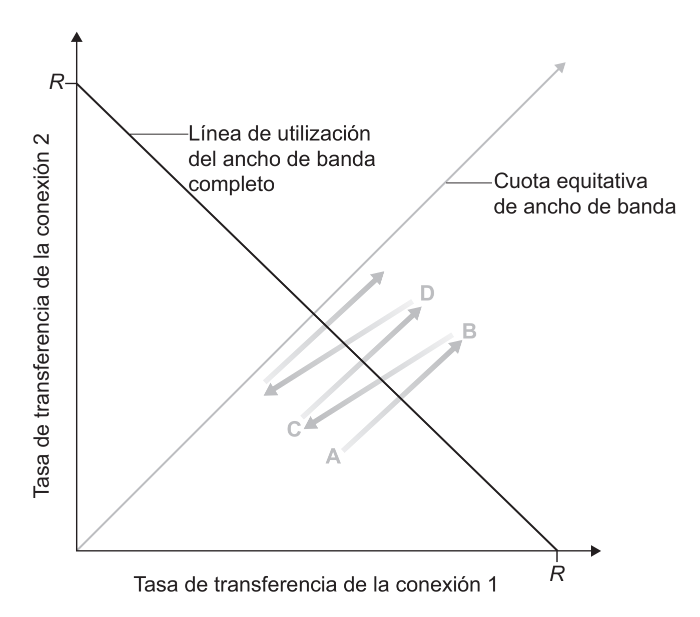


Figura 3.56 • Tasa de transferencia alcanzada por las conexiones TCP 1 y 2.

flecha que sale del origen con un ángulo de 45 grados (cuota equitativa de ancho de banda). Idealmente, la suma de las dos tasas de transferencia tiene que ser igual a *R* (evidentemente, que cada conexión reciba una cuota equitativa de la capacidad del enlace, pero igual a cero, no es una situación deseable). Por tanto, el objetivo debería ser que las tasas de transferencia alcanzadas se encuentren en algún punto próximo a la intersección de la línea de cuota equitativa de ancho de banda con la línea de utilización del ancho de banda completo mostradas en la Figura 3.56.

Suponga que los tamaños de ventana de TCP son tales que, en un instante determinado, las conexiones 1 y 2 alcanzan las tasas de transferencia indicadas por el punto *A* de la Figura 3.56. Puesto que la cantidad de ancho de banda del enlace conjunto consumido por las dos conexiones es menor que *R*, no se producirá ninguna pérdida y ambas conexiones incrementarán sus tamaños de ventana en 1 MSS por RTT como resultado del algoritmo de evitación de la congestión de TCP. Por tanto, la tasa de transferencia conjunta de las dos conexiones sigue la línea de 45 grados (incremento equitativo para ambas conexiones) partiendo del punto *A*. Finalmente, el ancho de banda del enlace consumido conjuntamente por las dos conexiones será mayor que *R,* por lo que terminará produciéndose una pérdida de paquetes. Suponga que las conexiones 1 y 2 experimentan una pérdida de paquetes cuando alcanzan las tasas de transferencia indicadas por el punto *B*. Entonces las conexiones 1 y 2 reducen el tamaño de sus ventanas en un factor de dos. Las tasas de transferencia resultantes se encuentran por tanto en el punto *C,* a medio camino de un vector que comienza en *B* y termina en el origen. Puesto que el ancho de banda conjunto utilizado es menor que *R* en el punto *C,* de nuevo las dos conexiones incrementan sus tasas de transferencia a lo largo de la línea de 45 grados partiendo de *C*. Finalmente, terminará por producirse de nuevo una pérdida de paquetes, por ejemplo, en el punto *D,* y las dos conexiones otra vez reducirán el tamaño de sus ventanas en un factor de dos, y así sucesivamente. Compruebe que el ancho de banda alcanzado por ambas conexiones fluctúa a lo largo de la línea que indica una cuota equitativa del ancho de banda. Compruebe también que las dos conexiones terminarán convergiendo a este comportamiento, independientemente de su posición inicial dentro de ese espacio bidimensional. Aunque en este escenario se han hecho una serie de suposiciones ideales, permite proporcionar una idea intuitiva de por qué TCP hace que el ancho de banda se reparta de forma equitativa entre las conexiones.

En este escenario ideal hemos supuesto que sólo las conexiones TCP atraviesan el enlace de cuello de botella, que las conexiones tienen el mismo valor de RTT y que sólo hay una conexión TCP asociada con cada pareja origen-destino. En la práctica, estas condiciones normalmente no se dan y las aplicaciones cliente-servidor pueden por tanto obtener cuotas desiguales del ancho de banda del enlace. En particular, se ha demostrado que cuando varias conexiones comparten un cuello de botella común, aquellas sesiones con un valor de RTT menor son capaces de apropiarse más rápidamente del ancho de banda disponible en el enlace, a medida que éste va liberándose (es decir, abren más rápidamente sus ventanas de congestión) y por tanto disfrutan de una tasa de transferencia más alta que aquellas conexiones cuyo valor de RTT es más grande [Lakshman 1997].

Equidad en UDP

Acabamos de ver cómo el control de congestión de TCP regula la velocidad de transmisión de una aplicación mediante el mecanismo de la ventana de congestión. Muchas aplicaciones multimedia, como las videoconferencias y la telefonía por Internet, a menudo no se ejecutan sobre TCP precisamente por esta razón (no desean que su velocidad de transmisión se

regule, incluso aunque la red esté muy congestionada). En lugar de ello, estas aplicaciones prefieren ejecutarse sobre UDP, que no incorpora un mecanismo de control de la congestión. Al ejecutarse sobre UDP, las aplicaciones pueden entregar a la red sus datos de audio y de vídeo a una velocidad constante y, ocasionalmente, perder paquetes, en lugar de reducir sus velocidades a niveles "equitativos" y no perder ningún paquete. Desde la perspectiva de TCP, las aplicaciones multimedia que se ejecutan sobre UDP no son equitativas (no cooperan con las demás conexiones ni ajustan sus velocidades de transmisión apropiadamente). Dado que el control de congestión de TCP disminuye la velocidad de transmisión para hacer frente a un aumento de la congestión (y de las pérdidas) y los orígenes de datos UDP no lo hacen, puede darse el caso de que esos orígenes UDP terminen por expulsar al tráfico TCP. Un área actual de investigación es el desarrollo de mecanismos de control de congestión para Internet que impidan que el tráfico UDP termine por reducir a cero la tasa de transferencia de Internet [Floyd 1999; Floyd 2000; Kohler 2006].

Equidad y conexiones TCP en paralelo

Pero aunque se pudiera forzar al tráfico UDP a comportarse equitativamente, el problema de la equidad todavía no estaría completamente resuelto. Esto es porque no hay nada que impida a una aplicación basada en TCP utilizar varias conexiones en paralelo. Por ejemplo, los navegadores web a menudo utilizan varias conexiones TCP en paralelo para transferir los distintos objetos contenidos en una página web (en la mayoría de los navegadores puede configurarse el número exacto de conexiones múltiples). Cuando una aplicación emplea varias conexiones en paralelo obtiene una fracción grande del ancho de banda de un enlace congestionado. Por ejemplo, considere un enlace cuya velocidad es R que soporta nueve aplicaciones entrantes cliente-servidor, utilizando cada una de las aplicaciones una conexión TCP. Si llega una nueva aplicación y también emplea una conexión TCP, entonces cada conexión tendrá aproximadamente la misma velocidad de transmisión de $R/10$. Pero si esa nueva aplicación utiliza 11 conexiones TCP en paralelo, entonces la nueva aplicación obtendrá una cuota no equitativa de más de $R/2$. Dado que el tráfico web es el dominante en Internet, las conexiones múltiples en paralelo resultan bastante comunes.

TP13

3.8 Resumen

Hemos comenzado este capítulo estudiando los servicios que un protocolo de la capa de transporte puede proporcionar a las aplicaciones de red. En uno de los extremos, el protocolo de capa de transporte puede ser muy simple y ofrecer un servicio poco sofisticado a las aplicaciones, poniendo a su disposición únicamente una función de multiplexación/demultiplexación para los procesos que se están comunicando. El protocolo UDP de Internet es un ejemplo de ese tipo de protocolo de la capa de transporte poco sofisticados. En el otro extremo, un protocolo de la capa de transporte puede proporcionar a las aplicaciones diversos tipos de garantías, como por ejemplo la de entrega fiable de los datos, garantías sobre los retardos y garantías concernientes al ancho de banda. De todos modos, los servicios que un protocolo de transporte puede proporcionar están a menudo restringidos por el modelo de servicio del protocolo subyacente de la capa de red. Si el protocolo de la capa de red no puede proporcionar garantías sobre los retardos o de ancho de banda a los segmentos de la capa de transporte, entonces el protocolo de la capa de transporte no puede proporcionar garantías de retardo ni de ancho de banda a los mensajes intercambiados por los procesos.

En la Sección 3.4 hemos visto que un protocolo de la capa de transporte puede proporcionar una transferencia fiable de los datos incluso aunque el protocolo de red subyacente sea no fiable. Allí vimos que que son muchas las sutilezas implicadas en la provisión de una transferencia fiable de los datos, pero que dicha tarea puede llevarse a cabo combinando cuidadosamente los paquetes de reconocimiento, los temporizadores, las retransmisiones y los números de secuencia.

Aunque hemos hablado de la transferencia fiable de los datos en este capítulo, debemos tener presente que esa transferencia fiable puede ser proporcionada por los protocolos de las capas de enlace, de red, de transporte o de aplicación. Cualquiera de las cuatro capas superiores de la pila de protocolos puede implementar los reconocimientos, los temporizadores, las retransmisiones y los números de secuencia, proporcionando así un servicio de transferencia de datos fiable a la capa que tiene por encima. De hecho, a lo largo de los años, los ingenieros e informáticos han diseñado e implementado de manera independiente protocolos de las capas de enlace, de red, de transporte y de aplicación que proporcionan una transferencia de datos fiable (aunque muchos de estos protocolos han desaparecido silenciosamente de la escena).

En la Sección 3.5 hemos examinado en detalle TCP, el protocolo fiable y orientado a la conexión de la capa de transporte de Internet. Hemos visto que TCP es bastante complejo, incluyendo técnicas de gestión de la conexión, de control de flujo y de estimación del tiempo de ida y vuelta, además de una transferencia fiable de los datos. De hecho, TCP es bastante más complejo de lo que nuestra descripción deja entrever; hemos dejado fuera de nuestra exposición, intencionadamente, diversos parches, correcciones y mejoras de TCP que están ampliamente implementadas en distintas versiones de dicho protocolo. De todos modos, todas estas complejidades están ocultas a ojos de la aplicación de red. Si el cliente en un host quiere enviar datos de forma fiable a un servidor implementado en otro host, se limita a abrir un socket TCP con el servidor y a bombear datos a través de dicho socket. Afortunadamente, la aplicación cliente-servidor es completamente inconsciente de toda la complejidad de TCP.

En la Sección 3.6 hemos examinado el control de congestión desde una perspectiva amplia, mientras que en la Sección 3.7 hemos mostrado cómo se implementa ese mecanismo de control de congestión en TCP. Allí vimos que el control de congestión es obligatorio parala buena saluda de la red. Sin él, la red puede colapsarse fácilmente, sin que al final puedan transportarse datos de terminal a terminal. En la Sección 3.7 vimos que TCP implementa un mecanismo de control de congestión terminal a terminal que incrementa de forma aditiva su tasa de transmisión cuando se evalúa que la ruta seguida por la conexión TCP está libre de congestión, mientras que esa tasa de transmisión se reduce multiplicativamente cuando se producen pérdidas de datos. Este mecanismo también trata de proporcionar a cada conexión TCP que pasa a través de un enlace congestionado una parte equitativa del ancho de banda del enlace. También hemos examinado con cierta profundidad el impacto que el establecimiento de la conexión TCP y el lento arranque de la misma tienen sobre la latencia. Hemos observado que, en muchos casos importantes, el establecimiento de la conexión y el arranque lento contribuyen significativamente al retardo terminal a terminal. Conviene recalcar una vez más que, aunque el control de congestión de TCP ha ido evolucionando a lo largo de los años, continúa siendo un área intensiva de investigación y es probable que continúe evolucionando en los próximos años.

El análisis realizado en este capítulo acerca de los protocolos específicos de transporte en Internet se ha centrado en UDP y TCP, que son los dos “caballos de batalla” de la capa de transporte de Internet. Sin embargo, dos décadas de experiencia con estos dos

protocolos han permitido identificar una serie de casos en los que ninguno de los dos resulta ideal. Los investigadores han dedicado, por tanto, grandes esfuerzos al desarrollo de protocolos adicionales de la capa de transporte, varios de los cuales son actualmente estándares propuestos por IETF.

El Protocolo de control de congestión para datagramas (DCCP, *Datagram Congestion Control Protocol*) [RFC 4340] proporciona un servicio no fiable con baja carga administrativa y orientado a mensajes, similar a UDP, pero que cuenta con un tipo de control de congestión seleccionado por la aplicación y que es compatible con TCP. Si una aplicación necesita una transferencia de datos fiable o semi-fiable, entonces el control de congestión sería implementado dentro de la propia aplicación, quizá utilizando los mecanismos que hemos estudiado en la Sección 3.4. DCCP está previsto para utilizarlo en aplicaciones tales como los flujos multimedia (véase el Capítulo 7) que pueden jugar con los compromisos existentes entre los requisitos de temporización y de fiabilidad en la entrega de datos, pero que quieran a la vez poder responder a situaciones de congestión en la red.

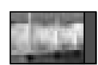
El Protocolo de transmisión para control de flujos (SCTP, *Stream Control Transmission Protocol*) [RFC 2960, RFC 3286] es un protocolo fiable orientado a mensajes, que permite multiplexar diferentes "flujos" de nivel de aplicación a través de una única conexión SCTP (una técnica conocida con el nombre de "multi-streaming"). Desde el punto de vista de la fiabilidad, los diferentes flujos que comparten la conexión se gestionan de forma separada, de modo que la pérdida de paquetes en uno de los flujos no afecte a la entrega de los datos en los otros. SCTP también permite transferir datos a través de dos rutas de salida cuando un host está conectado a dos o más redes; también existe la posibilidad de la entrega opcional de datos fuera de orden, así como otra serie de características interesantes. Los algoritmos de control de flujo y de control de congestión de SCTP son prácticamente los mismos que en TCP.

El protocolo de Control de tasa compatible con TCP (TFRC, *TCP-Friendly Rate Control*) [RFC 5348] es un protocolo de control de congestión más que un protocolo completo de la capa de transporte. TFRC especifica un mecanismo de control de congestión que podría ser utilizado en algún otro protocolo de transporte, como DCCP (de hecho, uno de los dos protocolos seleccionables por la aplicación existentes en DCCP es TFRC). El objetivo de TFRC es suavizar el comportamiento típico en "diente de sierra" (véase la Figura 3.54) que se experimenta en el control de congestión de TCP, al mismo tiempo que se mantiene una tasa de transmisión a largo plazo "razonablemente" próxima a la TCP. Con una tasa de transmisión de perfil más suave que TCP, TFRC está bien adaptado a aplicaciones multimedia tales como la telefonía IP o los flujos multimedia, en donde es importante mantener ese perfil suave de la tasa de transmisión. TFRC es un protocolo "basado en ecuaciones" que utiliza la tasa medida de pérdida de paquetes como entrada para una ecuación [Padhye 2000] que permite estimar cuál sería la tasa de transferencia TCP si una sesión TCP experimentara dicha tasa de pérdidas. Entonces, dicha tasa de transferencia se adopta como objetivo de tasa de transmisión para TFRC.

Sólo el futuro nos dirá si DCCP, SCTP o TFRC serán adoptados ampliamente o no. Aunque estos protocolos proporcionan claramente una serie de capacidades mejoradas respecto a TCP y UDP, estos dos protocolos han demostrado ser a lo largo de los años "lo suficientemente buenos". El que un "mejor" protocolos termine venciendo a otro que es "suficientemente bueno" dependerá de una compleja mezcla de aspectos técnicos, sociales y empresariales.

En el Capítulo 1 hemos visto que una red de computadoras puede dividirse entre lo que se denomina la "frontera de la red" y el "núcleo de la red". La frontera de la red cubre todo

lo que sucede en los sistemas terminales. Habiendo ya cubierto la capa de aplicación y la capa de transporte, nuestro análisis de la frontera de la red está completo, así que ha llegado el momento de explorar el núcleo de la red. Comenzaremos nuestro viaje en el siguiente capítulo, donde analizaremos la capa red, y seguiremos en el Capítulo 5 dedicado a la capa de enlace.



Problemas y cuestiones de repaso

Capítulo 3 Cuestiones de repaso

SECCIONES 3.1–3.3

R1. Suponga que la capa de red proporciona el siguiente servicio: la capa de red del host de origen acepta un segmento con un tamaño máximo de 1.200 bytes y una dirección de host de destino de la capa de transporte. La capa de red garantiza la entrega del segmento a la capa de transporte en el host de destino. Suponga que en el host de destino pueden ejecutarse muchos procesos de aplicaciones de red.

a. Diseñe el protocolo de la capa de transporte más simple posible que entregue los datos de la aplicación al proceso deseado en el host de destino. Suponga que el sistema operativo del host de destino ha asignado un número de puerto de 4 bytes a cada proceso de aplicación en ejecución.

b. Modifique este protocolo de manera que proporcione una "dirección de retorno" al proceso de destino.

c. En sus protocolos, ¿la capa de transporte "tiene que hacer algo" en el núcleo de la red de computadoras?

R2. Imagine una sociedad en la que todo el mundo perteneciera a una familia de seis miembros, todas las familias vivieran en su propia casa, cada casa tuviera una dirección única y cada persona de cada casa tuviera un nombre único. Imagine que esa sociedad dispone de un servicio de correos que transporta las cartas desde una vivienda de origen hasta una vivienda de destino. El servicio de correos requiere que (i) la carta se introduzca en un sobre y que (ii) la dirección de la casa de destino (y nada más) esté claramente escrita en el sobre. Suponga también que en cada familia hay un delegado que tiene asignada la tarea de recoger y distribuir las cartas a los restantes miembros de la familia. Las cartas no necesariamente proporcionan una indicación acerca de los destinatarios.

a. Partiendo de la solución del Problema R1, describa un protocolo que el delegado de la familia pueda utilizar para entregar las cartas de un miembro de la familia emisora a un miembro de la familia receptora.

b. En su protocolo, ¿el servicio de correos tienen que abrir el sobre y examinar la carta para proporcionar este servicio?

R3. Considere una conexión TCP entre el host A y el host B. Suponga que los segmentos TCP que viajan del host A al host B tienen un número de puerto de origen x y un número de puerto de destino y. ¿Cuáles son los números de puerto de origen y de destino para los segmentos que viajan del host B al host A?

R4. Describa por qué un desarrollador de aplicaciones puede decidir ejecutar una aplicación sobre UDP en lugar de sobre TCP.

R5. ¿Por qué razón el tráfico de voz y de vídeo suele enviarse sobre TCP en lugar de sobre UDP en la Internet de hoy día? (*Sugerencia*: la respuesta que estamos buscando no tiene nada que ver con el mecanismo de control de congestión de TCP.)

R6. ¿Es posible que una aplicación disfrute de una transferencia de datos fiable incluso si se ejecuta sobre UDP? En caso afirmativo, explique cómo.

R7. Sea un proceso del host C que tiene un socket UDP con el número de puerto 6789. Suponga también que los hosts A y B envían cada uno de ellos un segmento UDP al host C con el número de puerto de destino 6789. ¿Serán dirigidos ambos segmentos al mismo socket del host C? En caso afirmativo, ¿cómo sabrá el proceso del host C que estos dos segmentos proceden de dos hosts distintos?

R8. Suponga que un servidor web se ejecuta en el puerto 80 del host C. Suponga también que este servidor web utiliza conexiones persistentes y que actualmente está recibiendo solicitudes de dos hosts diferentes, A y B. ¿Están siendo enviadas todas las solicitudes al mismo socket del host C? Si están siendo pasadas a través de sockets diferentes, ¿utilizan ambos sockets el puerto 80? Explique y justifique su respuesta.

SECCIÓN 3.4

R9. En los protocolos `rdt` estudiados, ¿por qué necesitábamos introducir números de secuencia?

R10. En los protocolos `rdt` estudiados, ¿por qué necesitábamos introducir temporizadores?

R11. Suponga que el retardo de ida y vuelta entre el emisor y el receptor es constante y conocido por el emisor. ¿Se necesitaría en este caso un temporizador en el protocolo `rdt 3.0`, suponiendo que los paquetes pueden perderse? Explique su respuesta.

R12. Visite el applet de Java *Go-Back-N* en el sitio web del libro.

a. Haga que el emisor envíe cinco paquetes y luego detenga la animación antes de que cualquiera de los cinco paquetes alcance su destino. A continuación, elimine el primer paquete y reanude la animación. Describa lo que ocurre.

b. Repita el experimento, pero ahora deje que el primer paquete alcance su destino y elimine el primer paquete de reconocimiento. Describa lo que ocurre.

c. Para terminar, pruebe a enviar seis paquetes. ¿Qué ocurre?

R13. Repita el problema R12, pero ahora utilizando el applet de Java con repetición selectiva (SR). ¿En qué se diferencian los protocolos SR y GBN?

SECCIÓN 3.5

R14. ¿Verdadero o falso?

a. El host A está enviando al host B un archivo de gran tamaño a través de una conexión TCP. Suponga que el host B no tiene datos que enviar al host A. El host B no enviará paquetes de reconocimiento al host A porque el host B no puede superponer esos reconocimientos sobre los datos.

b. El tamaño de la ventana de recepción de TCP nunca varía mientras dura la conexión.

c. Suponga que el host A está enviando al host B un archivo de gran tamaño a través de una conexión TCP. El número de bytes no reconocidos que A envía no puede exceder el tamaño del buffer del receptor.

d. Suponga que el host A está enviando al host B un archivo de gran tamaño a través de una conexión TCP. Si el número de secuencia de un segmento en esta conexión es m, entonces el número de secuencia del siguiente segmento necesariamente tiene que ser $m + 1$.

e. El segmento TCP contiene un campo en su cabecera para `VentanaRecepcion`.

f. Suponga que el último `RTTMuestra` en una conexión TCP es igual a 1 segundo. El valor actual del `IntervaloFinDeTemporización` para la conexión será necesariamente   1 segundo.

g. Suponga que el host A envía al host B un segmento con el número de secuencia 38 y 4 bytes de datos a través de una conexión TCP. En este mismo segmento el número de reconocimiento necesariamente tiene que ser 42.

R15. Suponga que el host A envía dos segmentos TCP seguidos al host B a través de una conexión TCP. El primer segmento tiene el número de secuencia 90 y el segundo tiene el número de secuencia 110.

a. ¿Cuántos datos hay en el primer segmento?

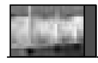
b. Suponga que el primer segmento se pierde pero el segundo llega a B. En el paquete de reconocimiento que el host B envía al host A, ¿cuál será el número de reconocimiento?

R16. Considere el ejemplo de la conexión Telnet de la Sección 3.5. Unos pocos segundos después de que el usuario escriba la letra 'C', escribe la letra 'R'. Después de escribir la letra 'R', ¿cuántos segmentos se envían y qué valores se almacenan en los campos número de secuencia y número de reconocimiento de los segmentos?

SECCIÓN 3.7

R17. Suponga que existen dos conexiones TCP en un cierto enlace de cuello de botella con una velocidad de R bps. Ambas conexiones tienen que enviar un archivo de gran tamaño (en la misma dirección a través del enlace de cuello de botella). Las transmisiones de los archivos se inician en el mismo instante. ¿Qué velocidad de transmisión podría proporcionar TCP a cada una de las conexiones?

R18. ¿Verdadero o falso? En el control de congestión de TCP, si el temporizador del emisor caduca, el valor de `umbral` se hace igual a la mitad de su valor anterior.



Problemas

P1. Suponga que el cliente A inicia una sesión Telnet con el servidor S. Aproximadamente en el mismo instante, el cliente B también inicia una sesión Telnet con el servidor S. Proporcione los posibles números de puerto de origen y de destino para:

a. Los segmentos enviados de A a S.
b. Los segmentos enviados de B a S.
c. Los segmentos enviados de S a A.
d. Los segmento enviados de S a B.
e. Si A y B son hosts diferentes, ¿es posible que el número de puerto de origen en los segmentos que van de A a S sea el mismo que en los segmentos que van de B a S?
f. ¿Qué ocurre si A y B son el mismo host?

P2. Considere la Figura 3.5. ¿Cuáles son los valores de los puertos de origen y de destino en los segmentos que fluyen desde el servidor de vuelta a los procesos cliente? ¿Cuáles son las direcciones IP de los datagramas de la capa de red que transportan los segmentos de la capa de transporte?

P3. UDP y TCP utilizan el complemento a 1 para calcular sus sumas de comprobación. Suponga que tiene los tres bytes de 8 bits siguientes: 01010011, 01010100, 01110100. ¿Cuál es el complemento a 1 de la suma de estos bytes? (Observe que aunque UDP y TCP utilizan palabras de 16 bits para calcular la suma de comprobación, en este problema le pedimos que considere sumas de 8 bits). Explique cómo funciona. ¿Por qué UDP utiliza el complemento a 1 de la suma; es decir, por qué no simplemente emplea la suma? Con el esquema del complemento a 1, ¿cómo detecta el receptor los errores? ¿Es posible que un error de un solo bit no sea detectado? ¿Qué ocurre si hay 2 bits erróneos?

P4. a. Suponga que tiene los 2 bytes siguientes: 01011100 y 01010110. ¿Cuál es el complemento a 1 de la suma de estos 2 bytes?

b. Suponga que tiene los 2 bytes siguientes: 11011010 y 00110110. ¿Cuál es el complemento a 1 de la suma de estos 2 bytes?

c. Para los bytes del apartado (a), proporcione un ejemplo en el que un bit cambie de valor en cada uno de los 2 bytes y aún así el complemento a 1 no varíe.

P5. Suponga que el receptor UDP calcula la suma de comprobación de Internet para el segmento UDP recibido y comprueba que se corresponde con el valor almacenado en el campo de suma de comprobación. ¿Puede el receptor estar completamente seguro de que no hay ningún bit erróneo? Explique su respuesta.

P6. Recuerde el motivo de corregir el protocolo `rdt2.1`. Demuestre que el receptor mostrado en la Figura 3.57 y el emisor mostrado en la Figura 3.11 pueden llegar a entrar en un estado de bloqueo tal que cada uno de ellos esté esperando a que se produzca un suceso que no ocurrirá nunca.

P7. En el protocolo `rdt3.0`, los paquetes ACK que fluyen del receptor al emisor no tienen números de secuencia (aunque tienen un campo ACK que contiene el número de secuencia del paquete que están reconociendo). ¿Por qué estos paquetes ACK no requieren números de secuencia?

P8. Dibuje la máquina de estados finitos correspondiente al lado receptor del protocolo `rdt3.0`.

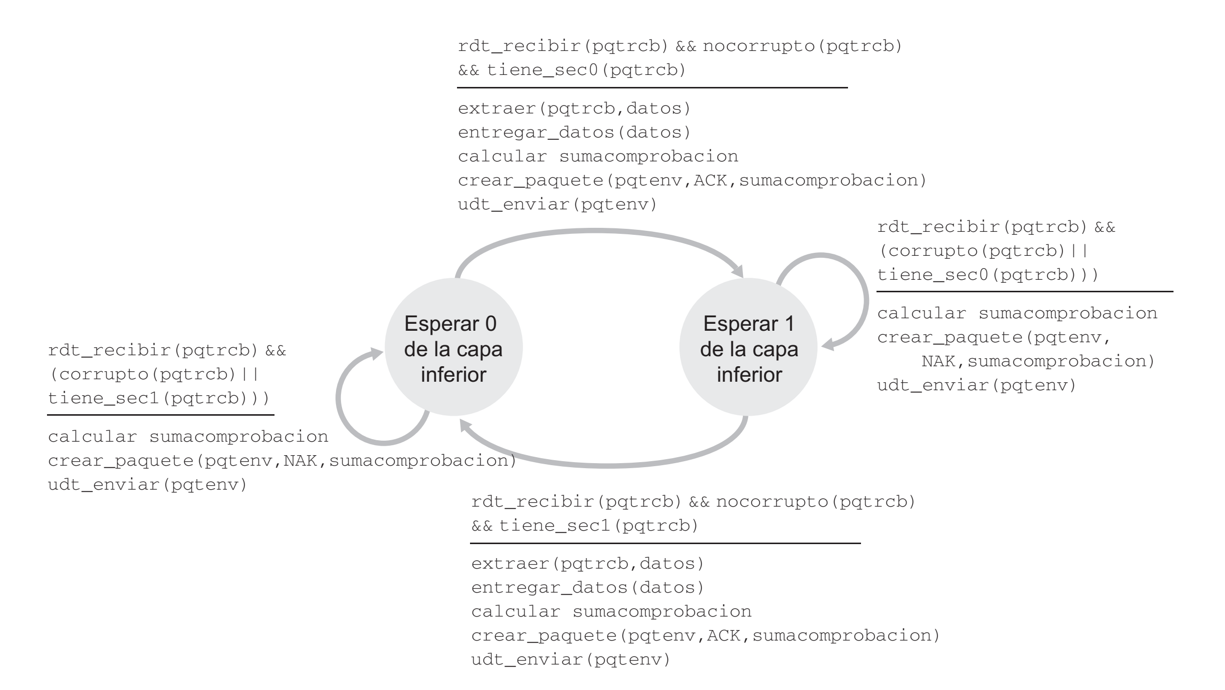


Figura 3.57 • Un receptor incorrecto para el protocolo `rdt 2.1`.

P9. Dibuje un esquema que muestre la operación del protocolo `rdt3.0` cuando los paquetes de datos y los paquetes de reconocimiento están corrompidos. Utilice un esquema similar al mostrado en la Figura 3.16.

P10. Sea un canal que puede perder paquetes pero del que se conoce su retardo máximo. Modifique el protocolo `rdt2.1` para incluir los fines de temporización y las retransmisiones del emisor. Argumente de manera informal por qué su protocolo puede comunicarse correctamente a través de este canal.

P11. El lado del emisor de `rdt3.0` simplemente ignora (es decir, no realiza ninguna acción) todos los paquetes recibidos que contienen un error o que presentan un valor erróneo en el campo número de reconocimiento (`acknum`) de un paquete de reconocimiento. Suponga que, en tales circunstancias, `rdt3.0` simplemente retransmite el paquete de datos actual. ¿Funcionaría en estas condiciones el protocolo? (*Sugerencia*: piense en lo que ocurriría si sólo hubiera errores de bit; no se producen pérdidas de paquetes pero sí pueden ocurrir sucesos de fin prematuro de la temporización. Considere cuántas veces se envía el paquete *n*, cuando *n* tiende a infinito.)

P12. Considere el protocolo `rdt 3.0`. Dibuje un diagrama que muestre que si la conexión de red entre el emisor y el receptor puede reordenar los mensajes (es decir, que dos mensajes que se propagan por el medio físico existente entre el emisor y el receptor pueden ser reordenados), entonces el protocolo de bit alternante no funcionará correctamente (asegúrese de identificar claramente el sentido en el que no funcionará correctamente). En el diagrama debe colocar el emisor a la izquierda y el receptor a la derecha, con el eje de tiempos en la parte inferior de la página

y deberá mostrar el intercambio de los mensajes de datos (D) y de reconocimiento (A). No olvide indicar el número de secuencia asociado con cualquier segmento de datos o de reconocimiento.

P13. Considere un protocolo de transferencia de datos fiable que sólo utiliza paquetes de reconocimiento negativo. Imagine que el emisor envía datos con muy poca frecuencia. ¿Sería preferible un protocolo con solo emplea paquetes NAK a uno que utilice paquetes ACK? ¿Por qué? Suponga ahora que el emisor tiene muchos datos que transmitir y que la conexión terminal a terminal experimenta muy pocas pérdidas. En este segundo caso, ¿sería preferible un protocolo que sólo emplee paquetes NAK a otro que utilice paquetes ACK? ¿Por qué?

P14. Considere el ejemplo mostrado en la Figura 3.17. ¿Cuál tiene que ser el tamaño de la ventana para que la tasa de utilización del canal sea mayor del 95 por ciento? Suponga que el tamaño de un paquete es de 1.500 bytes, incluyendo tanto los campos de cabecera como los datos.

P15. Suponga que una aplicación utiliza el protocolo `rdt 3.0` como su protocolo de la capa de transporte. Como el protocolo de parada y espera tiene una tasa de utilización del canal muy baja (como se ha demostrado en el ejemplo de conexión que atraviesa el país de costa a costa), los diseñadores de esta aplicación permiten al receptor devolver una serie (más de dos) de ACK 0 y ACK 1 alternantes incluso si los correspondientes datos no han llegado al receptor. ¿Debería este diseño aumentar la tasa de utilización del canal? ¿Por qué? ¿Existe algún problema potencial con esta técnica? Explique su respuesta.

P16. En el protocolo SR genérico que hemos estudiado en la Sección 3.4.4, el emisor transmite un mensaje tan pronto como está disponible (si se encuentra dentro de la ventana) sin esperar a recibir un paquete de reconocimiento. Suponga ahora que deseamos disponer de un protocolo SR que envíe mensajes de dos en dos. Es decir, el emisor enviará una pareja de mensajes y enviará la siguiente pareja de mensajes solo cuando sepa que los dos mensajes de la primera pareja se han recibido correctamente.

Suponga que el canal puede perder mensajes pero no corromperlos ni tampoco reordenarlos. Diseñe un protocolo de control de errores para un servicio de transferencia de mensajes fiable y unidireccional. Proporcione una descripción de las máquinas de estados finitos del emisor y del receptor. Describa el formato de los paquetes intercambiados por el emisor y el receptor. Si utiliza alguna llamada a procedimiento distinta de las empleadas en la Sección 3.4 (por ejemplo, `udt_enviar()`, `iniciar_temporizador()`, `rdt_recibir()`, etc.), defina claramente las acciones que realizan. Proporcione un ejemplo (una gráfica temporal del emisor y del receptor) que muestre cómo este protocolo se recupera de la pérdida de paquetes.

P17. Considere un escenario en el que el host A desea enviar simultáneamente paquetes a los hosts B y C. El host A está conectado a B y C a través de un canal de multidifusión (*broadcast*) (un paquete enviado por A es transportado por el canal tanto a B como a C). Suponga que el canal de multidifusión que conecta A, B y C puede perder y corromper de manera independiente los paquetes (es decir, puede ocurrir, por ejemplo, que un paquete enviado desde A llegue correctamente a B, pero no a C). Diseñe un protocolo de control de errores similar a un protocolo de parada y espera que permita

transferir paquetes de forma fiable de A a B y C, de manera que A no obtendrá nuevos datos de la capa superior hasta que separa que tanto B como C han recibido correctamente el paquete actual. Proporcione las descripciones de las máquinas de estados finitos de A y C. (*Sugerencia:* la FSM de B será prácticamente la misma que la de C.) Proporcione también una descripción del formato o formatos de paquete utilizados.

P18. Considere un escenario en el que el host A y el host B desean enviar mensajes al host C. Los hosts A y C están conectados mediante un canal que puede perder y corromper (pero no reordenar) los mensajes. Los hosts B y C están conectados a través de otro canal (independiente del canal que conecta a A y C) que tiene las mismas propiedades. La capa de transporte del host C tiene que alternar la entrega de los mensajes que A y B tienen que pasar a la capa superior (es decir, primero entrega los datos de un paquete de A y luego los datos de un paquete de B, y así sucesivamente). Diseñe un protocolo de control de errores de tipo parada y espera para transferir de forma fiable los paquetes de A y B a C, con una entrega alternante en el host C, como hemos descrito anteriormente. Proporcione las descripciones de las FSM de A y C. (*Sugerencia:* la FSM de B será prácticamente la misma que la de A.) Proporcione también una descripción del formato o formatos de paquete utilizados.

P19. Sea un protocolo GBN con una ventana de emisor de 3 y un rango de números de secuencia de 1.024. Suponga que en el instante t el siguiente paquete en orden que el receptor está esperando tiene el número de secuencia k. Suponga que el medio de transmisión no reordena los mensajes. Responda a las siguientes cuestiones:

a. ¿Cuáles son los posibles conjuntos de números de secuencia que pueden estar dentro de la ventana del emisor en el instante t? Justifique su respuesta.

b. ¿Cuáles son todos los valores posibles del campo ACK en todos los posibles mensajes que están actualmente propagándose de vuelta al emisor en el instante t? Justifique su respuesta.

P20. Suponga que tenemos dos entidades de red, A y B. B tiene que enviar a A un conjunto de mensajes de datos, cumpliendo los siguientes convenios. Cuando A recibe una solicitud de la capa superior para obtener el siguiente mensaje de datos (D) de B, A tiene que enviar un mensaje de solicitud (R) a B a través del canal que va de A a B. Sólo cuando B recibe un mensaje R puede devolver un mensaje de datos (D) a A a través del canal de B a A. A tiene que entregar exactamente una copia de cada mensaje D a la capa superior. Los mensajes R se pueden perder (pero no corromper) en el canal de A a B; los mensajes D, una vez enviados, siempre son correctamente entregados. El retardo a lo largo de ambos canales es desconocido y variable.

Diseñe (proporcione una descripción de la FSM de) un protocolo que incorpore los mecanismos apropiados para compensar las pérdidas del canal de A a B e implemente el paso de los mensajes a la capa superior de la entidad A, como se ha explicado anteriormente. Utilice sólo aquellos mecanismos que sean absolutamente necesarios.

P21. Considere los protocolos GBN y SR. Suponga que el tamaño del espacio de números de secuencia es k. ¿Cuál es la máxima ventana de emisor permitida que evitará la ocurrencia de problemas como los indicados en la Figura 3.27 para cada uno de estos protocolos?

P22. Responda verdadero o falso a las siguientes preguntas y justifique brevemente sus respuestas:

a. Con el protocolo SR, el emisor puede recibir un ACK para un paquete que se encuentra fuera de su ventana actual.

b. Con GBN, el emisor puede recibir un ACK para un paquete que se encuentra fuera de su ventana actual.

c. El protocolo de bit alternante es igual que el protocolo SR pero con un tamaño de ventana en el emisor y en el receptor igual a 1.

d. El protocolo de bit alternante es igual que el protocolo GBN pero con un tamaño de ventana en el emisor y en el receptor igual a 1.

P23. Hemos dicho que una aplicación puede elegir UDP como protocolo de transporte porque UDP ofrece a la aplicación un mayor grado de control (que TCP) en lo relativo a qué datos se envían en un segmento y cuándo.

a. ¿Por qué una aplicación tiene más control sobre qué datos se envían en un segmento?

b. ¿Por qué una aplicación tiene más control sobre cuándo se envía el segmento?

P24. Se desea transferir un archivo de gran tamaño de *L* bytes del host A al host B. Suponga un MSS de 536 bytes.

a. ¿Cuál es el valor máximo de *L* tal que los números de secuencia de TCP no se agoten? Recuerde que el campo número de secuencia de TCP tiene 4 bytes.

b. Para el valor de *L* que haya obtenido en el apartado (a), calcule el tiempo que tarda en transmitirse el archivo. Suponga que a cada segmento se añade un total de 66 bytes para la cabecera de la capa de transporte, de red y de enlace de datos antes de enviar el paquete resultante a través de un enlace a 155 Mbps. Ignore el control de flujo y el control de congestión de modo que A pueda bombear los segmentos seguidos y de forma continuada.

P25. Los hosts A y B están comunicándose a través de una conexión TCP y el host B ya ha recibido de A todos los bytes hasta el byte 126. Suponga que a continuación el host A envía dos segmentos seguidos al host B. El primer y el segundo segmentos contienen, respectivamente, 70 y 50 bytes de datos. En el primer segmento, el número de secuencia es 127, el número del puerto de origen es 302 y el número de puerto de destino es 80. El host B envía un paquete de reconocimiento cuando recibe un segmento del host A.

a. En el segundo segmento enviado del host A al B, ¿Cuáles son el número de secuencia, el número del puerto de origen y el número del puerto de destino?

b. Si el primer segmento llega antes que el segundo segmento, ¿cuál es el número de reconocimiento, el número del puerto de origen y el número del puerto de destino en el ACK correspondiente al primer segmento?

c. Si el segundo segmento llega antes que el primero, ¿cuál es el número de reconocimiento en el ACK correspondiente al primer segmento?

d. Suponga que los dos segmentos enviados por A llegan en orden a B. El primer paquete de reconocimiento se pierde y el segundo llega después de transcurrido el primer intervalo de fin de temporización. Dibuje un diagrama de temporización que

muestre estos segmentos y todos los restantes segmentos y paquetes de reconocimiento enviados. (Suponga que no se producen pérdidas de paquetes adicionales.) para cada uno de los segmentos que incluya en su diagrama, especifique el número de secuencia y el número de bytes de datos; para cada uno de los paquetes de reconocimiento que añada, proporcione el número de reconocimiento.

P26. Los hosts A y B están directamente conectados mediante un enlace a 100 Mbps. Existe una conexión TCP entre los dos hosts y el host A está transfiriendo al host B una archivo de gran tamaño a través de esta conexión. El host A puede enviar sus datos de la capa de aplicación a su socket TCP a una velocidad tan alta como 120 Mbps pero el host B sólo puede leer los datos almacenados en su buffer de recepción TCP a una velocidad máxima de 60 Mbps. Describa el efecto del control de flujo de TCP.

P27. En la Sección 3.5.6 se han estudiado las cookies SYN.

a. ¿Por qué es necesario que el servidor utilice un número de secuencia inicial especial en SYNACK?
b. Suponga que un atacante sabe que un host objetivo utiliza cookies SYN. ¿Puede el atacante crear conexiones semi-abiertas o completamente abiertas enviando simplemente un paquete ACK al host objetivo? ¿Por qué?
c. Suponga que un atacante recopila una gran cantidad de números de secuencia iniciales enviados por el servidor. ¿Puede el atacante hacer que el servidor cree muchas conexiones completamente abiertas enviando paquetes ACK con esos números de secuencia iniciales? ¿Por qué?

P28. Considere la red mostrada en el escenario 2 de la Sección 3.6.1. Suponga que ambos hosts emisores A y B tienen definidos valores de fin de temporización fijos.

a. Demuestre que aumentar el tamaño del buffer finito del router puede llegar a hacer que se reduzca la tasa de transferencia (λ_{out}).
b. Suponga ahora que ambos hosts ajustan dinámicamente su valores de fin de temporización (como lo hace TCP) basándose en el retardo del buffer del router. ¿Incrementar el tamaño del buffer ayudaría a incrementar la tasa de transferencia? ¿Por qué?

P29. Considere el procedimiento de TCP para estimar RTT. Suponga que $\alpha = 0,1$. Sea `RTTMuestra`$_1$ la muestra de RTT más reciente, `RTTMuestra`$_2$ la siguiente muestra de RTT más reciente, y así sucesivamente.

a. Para una conexión TCP determinada, suponga que han sido devueltos cuatro paquetes de reconocimiento con las correspondientes muestras de RTT, `RTTMuestra`$_4$, `RTTMuestra`$_3$, `RTTMuestra`$_2$ y `RTTMuestra`$_1$. Exprese `RTTEstimado` en función de las cuatro muestras de RTT.
b. Generalize la fórmula para *n* muestras de RTT.
c. En la fórmula del apartado (b), considere que *n* tiende a infinito. Explique por qué este procedimiento de cálculo del promedio se conoce como media móvil exponencial.

P30. En la Sección 3.5.3, se ha estudiado la estimación de RTT en TCP. ¿Por qué cree que TCP evita medir `RTTMuestra` para los segmentos retransmitidos?

P31. ¿Cuál es la relación entre la variable `EnviarBase` de la Sección 3.5.4 y la variable `UltimoByteRecibido` de la Sección 3.5.5?

P32. ¿Cuál es la relación entre la variable `UltimoByteRecibido` de la Sección 3.5.5 y la variable `y` de la Sección 3.5.4?

P33. En la Sección 3.5.4 hemos visto que TCP espera hasta que ha recibido tres ACK duplicados antes de realizar una retransmisión rápida. ¿Por qué cree que los diseñadores de TCP han decidido no realizar una retransmisión rápida después de recibir el primer ACK duplicado correspondiente a un segmento?

P34. Compare GBN, SR y TCP (sin paquetes ACK retardados). Suponga que los valores de fin de temporización de los tres protocolos son los suficientemente grandes como para que 5 segmentos de datos consecutivos y sus correspondientes ACK puedan ser recibidos (si no se producen pérdidas en el canal) por el host receptor (host B) y el host emisor host (host A), respectivamente. Suponga que el host A envía 5 segmentos de datos al host B y que el segundo segmento (enviado desde A) se pierde. Al final, los 5 segmentos de datos han sido recibidos correctamente por el host B.

a. ¿Cuántos segmentos ha enviado en total el host A y cuantos ACK ha enviado en total el host B? ¿Cuáles son sus números de secuencia? Responda a esta pregunta para los tres protocolos.

b. Si los valores de fin de temporización para los tres protocolos son mucho mayores que 5 RTT, ¿qué protocolo entregará correctamente los cinco segmentos de datos en el menor intervalo de tiempo?

P35. En la descripción de TCP de la Figura 3.53, el valor del umbral se define como `umbral=VentanaCongestion/2` en varios sitios y el valor de `umbral` se hace igual a la mitad del tamaño de la ventana cuando se produce un suceso de pérdida. ¿Tiene que ser la velocidad a la que el emisor está transmitiendo cuando se produce un suceso de pérdida aproximadamente igual a `VentanaCongestion` segmentos por RTT? Explique su respuesta. Si su respuesta es no, ¿puede sugerir una forma diferente en la que se podría fijar el valor de `umbral`?

P36. Considere la Figura 3.46(b). Si λ'_{in} aumenta por encima de $R/2$, ¿puede λ_{out} incrementarse por encima de $R/3$? Explique su respuesta. Considere ahora la Figura 3.46(c). Si λ'_{in} aumenta por encima de $R/2$, ¿puede λ_{out} aumentar por encima de R/4 suponiendo que un paquete será reenviado dos veces como media desde el router al receptor? Explique su respuesta.

P37. Considere la Figura 3.58.

Suponiendo que TCP Reno es el protocolo que presenta el comportamiento mostrado en la figura, responda a las siguientes preguntas. En todos los casos, deberá proporcionar una breve explicación que justifique su respuesta.

a. Identifique los intervalos de tiempo cuando TCP está operando en el modo de arranque lento.

b. Identifique los intervalos de tiempo cuando TCP está operando en el modo de evitación de la congestión.

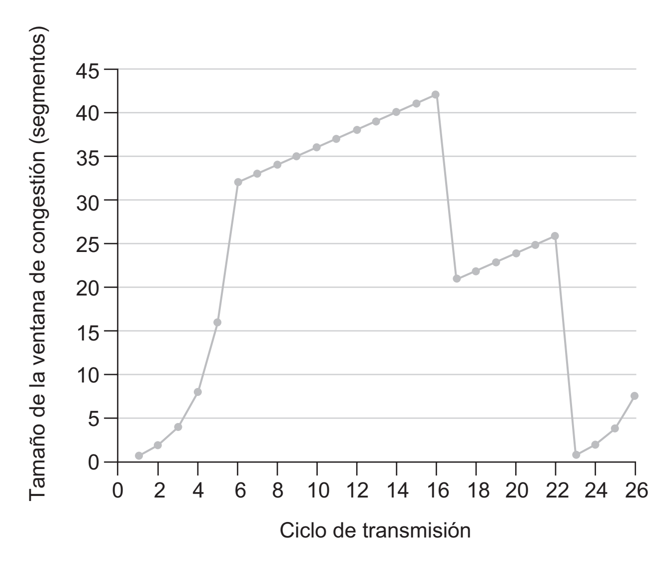


Figura 3.58 • Tamaño de ventana de TCP en función del tiempo.

c. Después del ciclo de transmisión 16, ¿se detecta la pérdida de segmento mediante tres ACK duplicados o mediante un fin de temporización?
d. Después del ciclo de transmisión 22, ¿se detecta la pérdida de segmento mediante tres ACK duplicados o mediante un fin de temporización?
e. ¿Cuál es el valor inicial de `umbral` en el primer ciclo de transmisión?
f. ¿Cuál es el valor de `umbral` transcurridos 18 ciclos de transmisión?
g. ¿Cuál es el valor de `umbral` transcurridos 24 ciclos de transmisión?
h. ¿Durante cuál ciclo de transmisión se envía el segmento 70?
i. Suponiendo que se detecta una pérdida de paquete después del ciclo de transmisión 26 a causa de la recepción de un triple ACK duplicado, ¿cuáles serán los valores del tamaño de la ventana de congestión y de `umbral`?
j. Suponga que se utiliza TCP Tahoe (en lugar de TCP Reno) y que se han recibido triples ACK duplicados en el ciclo de transmisión 16. ¿Cuáles serán los valores del tamaño de la ventana de congestión y de `umbral` en el ciclo de transmisión 19?
k. Suponga otra vez que se utiliza TCP Tahoe y que se produce un suceso de fin de temporización en el ciclo de transmisión 22. ¿Cuántos paquetes han sido enviados entre los ciclos de transmisión 17 a 22, ambos inclusive?

P38. Utilice la Figura 3.56, que ilustra la convergencia del algoritmo AIMD de TCP. Suponga que en lugar de un decrecimiento multiplicativo, TCP disminuye el tamaño de la ventana en una cantidad constante. ¿Convergería el algoritmo AIAD resultante hacia un algoritmo de cuota equitativa? Justifique su respuesta utilizando un diagrama similar al de la Figura 3.56.

P39. En la Sección 3.5.4, hemos explicado que el intervalo de fin de temporización se duplica después de un suceso de fin de temporización. Este mecanismos es una forma

de control de congestión. ¿Por qué TCP necesita un mecanismo de control de congestión basado en ventana (como hemos estudiado en la Sección 3.7) además de un mecanismo de duplicación del intervalo de fin de temporización?

P40. El host A está enviando un archivo de gran tamaño al host B a través de una conexión TCP. En esta conexión nunca se pierden paquetes y los temporizadores nunca caducan. La velocidad de transmisión del enlace que conecta el host A con Internet es R bps. Suponga que el proceso del host A es capaz de enviar datos a su socket TCP a una velocidad de S bps, donde $S = 10 \cdot R$. Suponga también que el buffer de recepción de TCP es lo suficientemente grande como para almacenar el archivo completo y que el buffer emisor sólo puede almacenar un porcentaje del archivo. ¿Qué impide al proceso del host A pasar datos de forma continua a su socket TCP a una velocidad de S bps? ¿El mecanismo de control de flujo de TCP, el mecanismo de control de congestión de TCP o alguna otra cosa? Razone su respuesta.

P41. Se envía un archivo de gran tamaño de un host a otro a través de una conexión TCP sin pérdidas.

a. Suponga que TCP utiliza el algoritmo AIMD para su control de congestión sin fase de arranque lento. Suponiendo que `VentanaCongestion` aumenta 1 MSS cada vez que se recibe un lote de paquetes ACK y suponiendo que los intervalos RTT son aproximadamente constantes, ¿Cuánto tiempo tarda `VentanaCongestion` en aumentar de 5 MSS a 11 MSS (si no se producen sucesos de pérdida de paquetes)?

b. ¿Cuál es la tasa de transferencia media (en función de MSS y RTT) para esta conexión hasta llegar al periodo RTT número 6?

P42. Recuerde la descripción macroscópica de la tasa de transferencia de TCP. En el periodo de tiempo que va desde que la velocidad de la conexión varía entre $W/(2 \cdot RTT)$ y W/RTT, sólo se pierde un paquete (justo al final del periodo).

a. Demuestre que la tasa de pérdidas (fracción de paquetes perdidos) es igual a:

$$L = \text{tasa de pérdidas} = \frac{1}{\frac{3}{8}W^2 + \frac{3}{4}W}$$

b. Utilice el resultado anterior para demostrar que si una conexión tiene una tasa de pérdidas igual a L, entonces su tasa promedio es aproximadamente igual a

$$\frac{1{,}22 \ MSS}{RTT\sqrt{L}}$$

P43. Considere una única conexión TCP (Reno) que emplea un enlace a 10Mbps que no almacena en buffer ningún dato. Suponga que este enlace es el único enlace congestionado entre los hosts emisor y receptor. Suponga también que el emisor TCP tiene que enviar al receptor un archivo de gran tamaño y que el buffer de recepción del receptor es mucho más grande que la ventana de congestión. Haremos además las siguientes suposiciones: el tamaño de segmento TCP es de 1.500 bytes, el retardo de propagación de ida y vuelta de esta conexión es igual a 100 milisegundos y esta conexión TCP siempre se encuentra en la fase de evitación de la congestión, es decir, ignoramos la fase de arranque lento.

a. ¿Cuál es el tamaño máximo de ventana (en segmentos) que esta conexión TCP puede alcanzar?

b. ¿Cuáles son el tamaño medio de ventana (en segmentos) y la tasa de transferencia media (en bps) de esta conexión TCP?

c. ¿Cuánto tiempo tarda esta conexión TCP en alcanzar de nuevo su tamaño de ventana máximo después de recuperarse de una pérdida de paquete?

P44. Continuando con el escenario descrito en el problema anterior, suponga que el enlace a 10Mbps puede almacenar en buffer un número finito de segmentos. Razone por qué para que el enlace esté siempre ocupado enviando datos, deberíamos seleccionar un tamaño de buffer que sea al menos igual al producto de la velocidad del enlace *C* y el retardo de propagación de ida y vuelta entre el emisor y el receptor.

P45. Repita el Problema 43, pero sustituyendo el enlace a 10 Mbps por un enlace a 10 Gbps. Observe que en la respuesta al apartado (c) habrá demostrado que se tarda mucho tiempo en que el tamaño de la ventana de congestión alcance su máximo después de recuperarse de una pérdida de paquete. Diseñe una solución que resuelva este problema.

P46. Sea *T* (medido en RTT) el intervalo de tiempo que una conexión TCP tarda en aumentar el tamaño de su ventana de congestión de *W*/2 a *W*, donde *W* es el tamaño máximo de la ventana de congestión. Demuestre que *T* es una función de la tasa de transferencia media de TCP.

P47. Considere un algoritmo AIMD de TCP simplificado en el que el tamaño de la ventana de congestión se mide en número de segmentos, no en bytes. En la fase de incremento aditivo, el tamaño de la ventana de congestión se incrementa en un segmento cada RTT. En la fase de decrecimiento multiplicativo, el tamaño de la ventana de congestión se reduce a la mitad (si el resultado no es un entero, redondee al entero más próximo). Suponga que dos conexiones TCP, C_1 y C_2, comparten un enlace congestionado cuya velocidad es de 30 segmentos por segundo. Suponemos que tanto C_1 como C_2 están en l fase de evitación de la congestión. El intervalo RTT de la conexión C_1es igual a 100 milisegundos y el de la conexión C_2 es igual a 200 milisegundos. Suponemos que cuando la velocidad de los datos en el enlace excede la velocidad del enlace, todas las conexiones TCP experimentan pérdidas de segmentos de datos.

a. Si en el instante t_0 el tamaño de la ventana de congestión de ambas conexiones, C_1 y C_2, es de 10 segmentos, ¿cuáles serán los tamaños de dichas ventanas de congestión después de transcurridos 2200 milisegundos?

b. ¿Obtendrán estas dos conexiones, a largo plazo, la misma cuota de ancho de banda del enlace congestionado? Explique su respuesta.

P48. Continúe con la red descrita en el problema anterior, pero ahora suponga que las dos conexiones TCP, C1 y C2, tienen el mismo intervalo RTT de 100 milisegundos. Suponga que en el instante t_0, el tamaño de la ventana de congestión de C1 es de 15 segmentos pero el tamaño de la ventana de congestión de C2 es igual a 10 segmentos.

a. ¿Cuáles serán los tamaños de las ventanas de congestión después de transcurridos 2200 milisegundos?

b. ¿Obtendrán estas dos conexiones, a largo plazo, la misma cuota de ancho de banda del enlace congestionado?

c. Decimos que dos conexiones están sincronizadas si ambas conexiones alcanzan su tamaño de ventana máximo al mismo tiempo y alcanzan su tamaño mínimo de ventana también al mismo tiempo. ¿Terminarán con el tiempo sincronizándose estas dos conexiones? En caso afirmativo, ¿cuáles son sus tamaños máximos de ventana?

d. ¿Ayudará esta sincronización a mejorar la tasa de utilización del enlace compartido? ¿Por qué? Esboce alguna idea para evitar esta sincronización.

P49. Veamos una modificación del algoritmo de control de congestión de TCP. En lugar de utilizar un incremento aditivo podemos emplear un incremento multiplicativo. Un emisor TCP incrementa su tamaño de ventana según una constante pequeña positiva a ($0 < a < 1$) cuando recibe un ACK válido. Halle la relación funcional existente entre la tasa de pérdidas L y el tamaño máximo de la ventana de congestión W. Demuestre que para esta conexión TCP modificada, independientemente de la tasa media de transferencia de TCP, una conexión TCP siempre invierte la misma cantidad de tiempo en incrementar el tamaño de su ventana de congestión de $W/2$ a W.

P50. En nuestra exposición sobre el futuro de TCP de la Sección 3.7 hemos destacado que para alcanzar una tasa de transferencia de 10 Gbps, TCP sólo podría tolerar una probabilidad de pérdida de segmentos de $2 \cdot 10^{-10}$ (o lo que es equivalente, un suceso de pérdida por cada 5.000.000.000 segmentos). Indique de dónde se obtienen los valores $2 \cdot 10^{-10}$ y 1 por cada 5.000.000 para los valores de RTT y MSS dados en la Sección 3.7. Si TCP tuviera que dar soporte a una conexión a 100 Gbps, ¿qué tasa de pérdidas sería tolerable?

P51. En nuestra exposición sobre el control de congestión de TCP de la Sección 3.7, implícitamente hemos supuesto que el emisor TCP siempre tiene datos que enviar. Consideremos ahora el caso en que el emisor TCP envía una gran cantidad de datos y luego en el instante t_1 se queda inactivo (puesto que no tiene más datos que enviar). TCP permanece inactivo durante un periodo de tiempo relativamente largo y en el instante t_2 quiere enviar más datos. ¿Cuáles son las ventajas y las desventajas de que TCP tengan que utilizar los valores de `VentanaCongestion` y `umbral` de t_1 cuando comienza a enviar datos en el instante t_2? ¿Qué alternativa recomendaría? ¿Por qué?

P52. En este problema vamos a investigar si UDP o TCP proporcionan un cierto grado de autenticación del punto terminal.

a. Considere un servidor que recibe una solicitud dentro de un paquete UDP y responde a la misma dentro de un paquete UDP (por ejemplo, como en el caso de un servidor DNS). Si un cliente con la dirección IP X suplanta su dirección con la dirección Y, ¿A dónde enviará el servidor su respuesta?

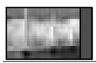
b. Suponga que un servidor recibe un SYN con la dirección IP de origen Y, y después de responder con un SYNACK, recibe un ACK con la dirección IP de origen Y y con el número de reconocimiento correcto. Suponiendo que el servidor elige un número de secuencia inicial aleatorio y que no existe ningún atacante interpuesto (*man-in-the-middle*), ¿puede el servidor estar seguro de que el cliente está en la dirección Y (y no en alguna otra dirección X que esté intentando suplantar a Y)?

P53. En este problema, vamos a considerar el retardo introducido por la fase de arranque lento de TCP. Se tiene un cliente y un servidor web directamente conectados mediante un enlace a velocidad R. Suponga que el cliente desea extraer un objeto cuyo tamaño es exactamente igual a 15 S, donde S es el tamaño máximo de segmento (MSS). Sea RTT el tiempo de transmisión de ida y vuelta entre el cliente y el servidor (suponemos que es constante). Ignorando las cabeceras del protocolo, determine el tiempo necesario para recuperar el objeto (incluyendo el tiempo de establecimiento de la conexión TCP) si

a. $4\,S/R > S/R + RTT > 2S/R$

b. $S/R + RTT > 4\,S/R$

c. $S/R > RTT$.



Preguntas para la discusión

D1. ¿Qué es el secuestro de una conexión TCP? ¿Cómo se puede hacer?

D2. En la Sección 3.7 hemos dicho que una aplicación cliente-servidor puede crear "de forma no equitativa" muchas conexiones simultáneas en paralelo. ¿Qué se puede hacer para que Internet sea realmente una red equitativa?

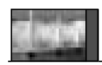
D3. Consulte la literatura de investigación para ver a qué se refiere el concepto de *orientado a TCP (TCP friendly)*. Lea también la entrevista a Sally Floyd al final del capítulo. Describa en una página el concepto de orientación a TCP.

D4. Al final de la Sección 3.7.1 hemos abordado el hecho de que una aplicación puede abrir varias conexiones TCP y conseguir una tasa de transferencia más alta (o lo que es lo mismo, una velocidad de transferencia de datos más rápida). ¿Qué ocurriría si todas las aplicaciones intentaran mejorar su rendimiento utilizando varias conexiones? ¿Cuáles son algunas de las dificultades derivadas de hacer que un elemento de red determine si una aplicación está utilizando varias conexiones TCP?

D5. Además de la exploración de puertos TCP y UDP, ¿de qué funcionalidad dispone nmap? Recopile trazas de los intercambios de paquetes de nmap mediante Ethereal (o cualquier otro analizador de paquetes, *sniffer*). Utilice las trazas para explicar cómo operan algunas de las funcionalidades avanzadas.

D6. En la descripción de TCP proporcionada en la Figura 3.53, el valor inicial de `VentanaCongestion` es 1. Consulte los libros disponibles y el RFC de Internet y comente algunas de las técnicas alternativas que se han propuesto para establecer el valor inicial de `VentanaCongestion`.

D7. Consulte la documentación disponible sobre SCTP [RFC 2960, RFC 3286]. ¿Cuáles son las aplicaciones que los diseñadores prevén para SCTP? ¿Qué funcionalidades de SCTP se añadieron para satisfacer las necesidades de esas aplicaciones?



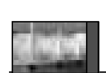
Tareas de programación

Implementación de un protocolo de transporte fiable

En esta tarea de programación tendrá que escribir el código para la capa de transporte del emisor y el receptor con el fin de implementar un protocolo de transferencia de datos fiable simple. Hay disponibles dos versiones de esta práctica de laboratorio: la versión del protocolo de bit alternante y la versión del protocolo GBN. Esta práctica de laboratorio le resultará entretenida y su implementación diferirá muy poco de lo que se necesita en una situación real.

Puesto que probablemente no dispone de máquinas autónomas (con un sistema operativo que pueda modificar), su código tendrá que ejecutarse en un entorno simulado hardware/software. Sin embargo, la interfaz de programación proporcionada a sus rutinas (el código que efectuará las llamadas a sus entidades desde las capas superior e inferior) es muy similar a la que se utiliza en un entorno UNIX real. (De hecho, las interfaces software descritas en esta tarea de programación son mucho más realistas que los emisores y receptores con bucles infinitos que se describen en muchos textos.) También se simula el arranque y la detención de temporizadores, y las interrupciones de los temporizadores harán que se active su rutina de tratamiento de temporizadores.

La tarea completa de laboratorio, así como el código que tendrá que compilar con su propio código está disponible en el sitio web del libro en http://www.awl.com/kurose-ross.



Práctica de laboratorio con Wireshark: exploración de TCP

En esta práctica de laboratorio tendrá que utilizar su navegador web para acceder a un archivo almacenado en un servidor web. Como en las prácticas de laboratorio con Wireshark anteriores, tendrá que utilizar Wireshark para capturar los paquetes que lleguen a su computadora. A diferencia de las prácticas anteriores, *también* podrá descargar una traza de paquetes (que Wireshark puede leer) del servidor web del que haya descargado el archivo. En esta traza del servidor encontrará los paquetes que fueron generados a causa de su propio acceso al servidor web. Analizará las trazas del lado del cliente y de lado del servidor para explorar los aspectos de TCP. En particular, tendrá que evaluar el rendimiento de la conexión TCP entre su computadora y el servidor web. Tendrá que trazar el comportamiento de la ventana de TCP e inferirá la pérdida de paquetes, las retransmisiones, el comportamiento del control de flujo y del control de congestión y el tiempo de ida y vuelta estimado.

Como con el resto de las prácticas de laboratorio con Wireshark, la descripción completa de esta práctica está disponible en el sitio web del libro en http://www.awl.com/kurose-ross.

Práctica de laboratorio con Wireshark: exploración de UDP

En esta corta práctica de laboratorio, realizará una captura y un análisis de paquetes de su aplicación favorita que utilice UDP (por ejemplo, DNS o una aplicación multimedia como Skype). Como hemos visto en la Sección 3.3, UDP es un protocolo de transporte simple. En esta práctica de laboratorio tendrá que investigar los campos de cabecera del segmento UDP, así como el cálculo de la suma de comprobación.

Al igual que con todas las demás prácticas de laboratorio de Wireshark, la descripción completa de la práctica está disponible en el sitio web del libro en http://www.awl.com/kurose-ross.

Sally Floyd



Sally Floyd es investigadora en el Centro ICSI para investigación sobre Internet, un instituto dedicado a los temas de Internet y de la comunicación por red. Es bastante conocida en el sector por sus trabajos en el diseño de protocolos de Internet, en particular los relativos a las comunicaciones multicast fiables, al control de congestión (TCP), a la planificación de paquetes (RED) y al análisis de protocolos. Sally se graduó en Sociología en la universidad de California, Berkeley, y se licención y doctoró en Ciencias de la Computación en la misma universidad.

¿Por qué decidió estudiar Ciencias de la Computación?

Después de obtener mi grado en Sociología, tuve que decidir cómo ganarme la vida y acabé obteniendo un certificado de estudios de dos años en electrónica en un centro local de formación, después de lo cual pasé diez años trabajando en el campo de la electrónica y de la informática. Esto incluyó ocho años como ingeniero de sistemas de computadoras para los equipos que controlaban los trenes de Bay Area Rapid Transit. Después, decidí aprender algo más de informática formal y cursé la licenciatura en el Departamento de Ciencias de la Computación de UC Berkeley.

¿Por qué decidió especializarse en redes?

Durante la licenciatura comencé a interesarme por la informática teórica. Primero trabajé en el análisis probabilístico de algoritmos y luego en la teoría computacional del aprendizaje. También estaba trabajando en LBL (Lawrence Berkeley Laboratory) un día al mes y mi despacho estaba muy próxima a la de Van Jacobson, que por aquel entonces estaba trabajando en algoritmos de control de congestión para TCP. Van me preguntó si me gustaría trabajar durante el verano analizando una serie de algoritmos para un problema de red que implicaba la sincronización no deseada de mensajes periódicos de enrutamiento. Me pareció interesante y a eso es a lo que me dediqué durante el verano.

Después de terminar mi tesis doctoral, Van me ofreció un trabajo a tiempo completo para continuar con las investigaciones en el campo de las redes. Yo no tenía planeado pasarme tantos años trabajando en el teme de redes, pero para mí la investigación en el campo de las redes es más gratificante que la informática teórica. Me siento más satisfecha trabajando en el mundo de la ciencia aplicada, en el que las consecuencias de mi trabajo son más tangibles.

¿Cuál fue su primer trabajo en la industria informática? ¿A qué se dedicaba?

Mi primer empleo fue en BART (Bay Area Rapid Transit), de 1975 a 1982, trabajando en las computadoras que controlan los trenes de BART. Comencé como técnico, manteniendo t reparando los diversos sistemas de computadoras distribuidos que están implicados en el control del sistema BART.

Esto incluía un sistema central de computadoras y un sistema distribuido de minicomputadoras para el control del movimiento de los trenes, un sistema de computadoras DEC para la visualización de anuncios y de los destinos de los trenes en los paneles de nuncios y un sistema de computadoras Modcomp para recopilar información de los tornos de entrada a las estaciones. Mis últimos años en

BART los pasé trabajando en un proyecto conjunto BART/LBL para diseñar el sustituto del ya bastante antiguo sistema de computadoras de control de trenes de BART.

¿Cuál es la parte más atractiva de su trabajo?

La tarea de investigación es la que más atrae. Uno de los problemas de investigación incluye explorar los problemas futuros relacionados con el control de congestión, para aplicaciones tales como los flujos multimedia. Un segundo tema es el de resolver los impedimentos existentes en las redes para poder implementar una comunicación más explícita entre los routers y los nodos terminales. Estos impedimentos pueden incluir los túneles IP y las rutas MPLS, la existencia de routers o equipos intermedios que eliminan paquetes que contienen opciones IP, la existencia de redes complejas de la capa 2 y las potenciales debilidades que hacen posibles los ataques a la red. Un tercer tema bastante activo consiste en explorar cómo la elección de modelos de escenarios para el análisis, la simulación y la experimentación afecta a nuestra evaluación del rendimiento de los mecanismos del control de congestión. Puede encontrar más información acerca de estos temas en las páginas web dedicadas a DCCP, Quick-Start y TMRG, a las que puede acceder a través de la dirección http://www.icir.org/floyd.

¿Cuál prevé que sea el futuro de las redes y de Internet?

Una posibilidad es que las congestiones típicas con las que se encuentra el tráfico de Internet pasen a ser menos problemáticas a medida que el ancho de banda disponible se incremente a un ritmo más rápido que la demanda. La tendencia que preveo es hacia un nivel de congestión menos severo, aunque tampoco es imposible que a medio plazo nos encontremos con un futuro de congestión creciente, con ocasionales colapsos debidos a la congestión.

El futuro de la propia Internet o de la arquitectura Internet no lo tengo nada claro. Hay muchos factores que contribuyen a la rápida evolución de Internet, así que resulta difícil predecir cómo será la evolución de la red o de la arquitectura de red, o incluso predecir si esta evolución tendrá éxito a la hora de evitar los numerosos problemas potenciales que se irán presentando a lo largo del camino.

Una tendencia negativa bien conocida es la creciente dificultad de realizar cambios en la arquitectura de Internet. La arquitectura de Internet ya no es un todo coherente, y los diversos componentes como protocolos de transporte, mecanismos de enrutamiento, cortafuegos, equilibradores de carga, mecanismos de seguridad, etc., en ocasiones trabajan teniendo objetivos contrapuestos.

¿Qué personas le han inspirado profesionalmente?

Richard Karp, mi director de tesis doctoral, me enseño lo fundamental de cómo se lleva a cabo una investigación, mientras que Van Jacobson, mi “jefe de grupo” en LBL, es el responsable de que en mí se desarrollara un interés por las redes, así como de mi comprensión acerca de la infraestructura Internet. Dave Clark también me ha servido de inspiración, gracias a su clara visión de la arquitectura de Internet y a su papel en el desarrollo de esa arquitectura mediante sus investigaciones, escritos y participación en el IETF y otros foros de carácter público. Deborah Estrin también ha sido una fuente de inspiración gracias a su capacidad de concentración a a su efectividad , así como a su habilidad para tomar decisiones meditadas acerca de en qué va a trabajar y por qué.

Una de las razones de haber disfrutado trabajando en el área de la investigación acerca de las redes es, precisamente, que existen muchas personas en este sector a las que aprecio, respeto y admiro. Son inteligentes, trabajan duro, están muy comprometidas con el desarrollo de Internet y pueden ser una excelente compañía a la hora de ir a tomar una cerveza o de mantener una discusión amistosa después de un día de reuniones.



CAPÍTULO 4

La capa de red

Hemos estudiado en el capítulo anterior que la capa de transporte proporciona varias formas de comunicación proceso a proceso basándose en el servicio de comunicación host a host de la capa de red. También hemos visto que la capa de transporte lleva a cabo esta tarea sin saber cómo la capa de red implementa realmente este servicio. Así que es posible que se esté preguntando, ¿cuál es el mecanismo subyacente al servicio de comunicación de host a host que lo hace funcionar?

En este capítulo vamos a ver exactamente cómo la capa de red implementa el servicio de comunicación host a host. Veremos que, a diferencia de la capa de transporte, existe un componente de la capa de red en todos y cada uno de los hosts y routers de la red. Por esta razón, los protocolos de la capa de red se encuentran entre los más desafiantes (y, por tanto, entre los más interesantes) de la pila de protocolos.

La capa de red también es una de las capas más complejas de la pila de protocolos y, por tanto, serán muchas las cuestiones que vamos a tener que abordar. Comenzaremos nuestro estudio con una introducción a la capa de red y a los servicios que puede proporcionar. A continuación, volveremos sobre las dos técnicas empleadas en la estructuración del servicio de entrega de paquetes de la capa de red (los datagramas y el modelo de circuito virtual), de las que hablamos por primera vez en el Capítulo 1. Asimismo, veremos el papel fundamental que desempeña el direccionamiento en la entrega de paquetes a un host de destino.

En este capítulo haremos una importante distinción entre las funciones de **reenvío (*forwarding*)** y de **enrutamiento (*routing*)** de la capa de red. El reenvío implica la transferencia de un paquete desde un enlace de entrada a un enlace de salida dentro de un *mismo* router. El enrutamiento implica a *todos* los routers de una red, cuyas interacciones colectivas mediante los protocolos de enrutamiento determinan las rutas que seguirán los paquetes en sus viajes desde el origen hasta el destino. Ésta es una importante distinción que deberá tener presente a medida que avancemos en este capítulo.

Con el fin de profundizar en nuestros conocimientos acerca del reenvío de paquetes, vamos a echar un vistazo al “interior” de un router (a su organización y su arquitectura hard-

ware). A continuación, nos ocuparemos del reenvío de paquetes en Internet, junto con el célebre Protocolo de Internet (IP). Estudiaremos el direccionamiento de la capa de red y el formato de los datagramas IPv4. Después, exploraremos la Traducción de direcciones de red (NAT, *Network Address Translation*), la fragmentación de datagramas, el Protocolo de mensajes de control de Internet (ICMP, *Internet Control Message Protocol*) e IPv6.

Después volveremos a poner nuestra atención en la función de enrutamiento de la capa de red. Veremos que el trabajo de un protocolo de enrutamiento es determinar las mejores rutas que van de los emisores a los receptores. En primer lugar, estudiaremos la teoría de los algoritmos de enrutamiento, concentrándonos en las dos clases de algoritmos más importantes: el algoritmo de estado de enlaces y el algoritmo de vector de distancias. Dado que la complejidad de los algoritmos de enrutamiento crece considerablemente a medida que aumenta el número de routers de red, las técnicas de enrutamiento jerárquico también serán de nuestro interés. A continuación, veremos cómo llevar la teoría a la práctica al ocuparnos de los protocolos de enrutamiento internos de los sistemas autónomos de Internet (RIP, OSPF e IS-IS) y de su protocolo de enrutamiento entre sistemas autónomos, BGP. Terminaremos este capítulo con una exposición acerca del enrutamiento por difusión (*broadcast*) y por multidifusión (*multicast*).

En resumen, este capítulo consta de tres grandes partes. La primera abarca las Secciones 4.1 y 4.2 y se ocupa de las funciones y servicios de la capa de red. La segunda parte abarca las Secciones 4.3 y 4.4 y cubre la función de reenvío y, por último, la tercera parte que consta de las Secciones 4.5 a 4.7 se ocupa del enrutamiento.

4.1 Introducción

La Figura 4.1 muestra una red simple formada por dos hosts, H1 y H2, y varios routers en la ruta que va de H1 a H2. Supongamos que H1 está enviando información a H2; veamos entonces el papel de la capa de red en estos hosts y en los routers intervinientes. La capa de red en H1 toma segmentos de la capa de transporte en H1, encapsula cada segmento en un datagrama (es decir, un paquete de la capa de red) y, a continuación, envía los datagramas al router más próximo, R1. En el host de recepción, H2, la capa de red recibe los datagramas de su router más próximo R2, extrae los segmentos de la capa de transporte y los entrega a la capa de transporte de H2. La función principal de los routers es reenviar los datagramas desde los enlaces de entrada a los enlaces de salida. Observe que los routers de la Figura 4.1 se ilustran con una pila de protocolos truncada, es decir, sin capas por encima de la capa de red, porque (excepto para propósitos de control) los routers no ejecutan protocolos de la capa de transporte ni de la capa de aplicación como los que hemos examinado en los Capítulos 2 y 3.

4.1.1 Reenvío y enrutamiento

La función de la capa de red es por tanto tremendamente simple: transporta paquetes desde un host emisor a un host receptor. En la realización de esta tarea podemos identificar dos importantes funciones de la capa de red:

- *Reenvío (forwarding).* Cuando un paquete llega al enlace de entrada de un router, éste tiene que pasar el paquete al enlace de salida apropiado. Por ejemplo, un paquete que llega procedente de H1 al router R1 debe ser reenviado al siguiente router de la ruta hacia H2.

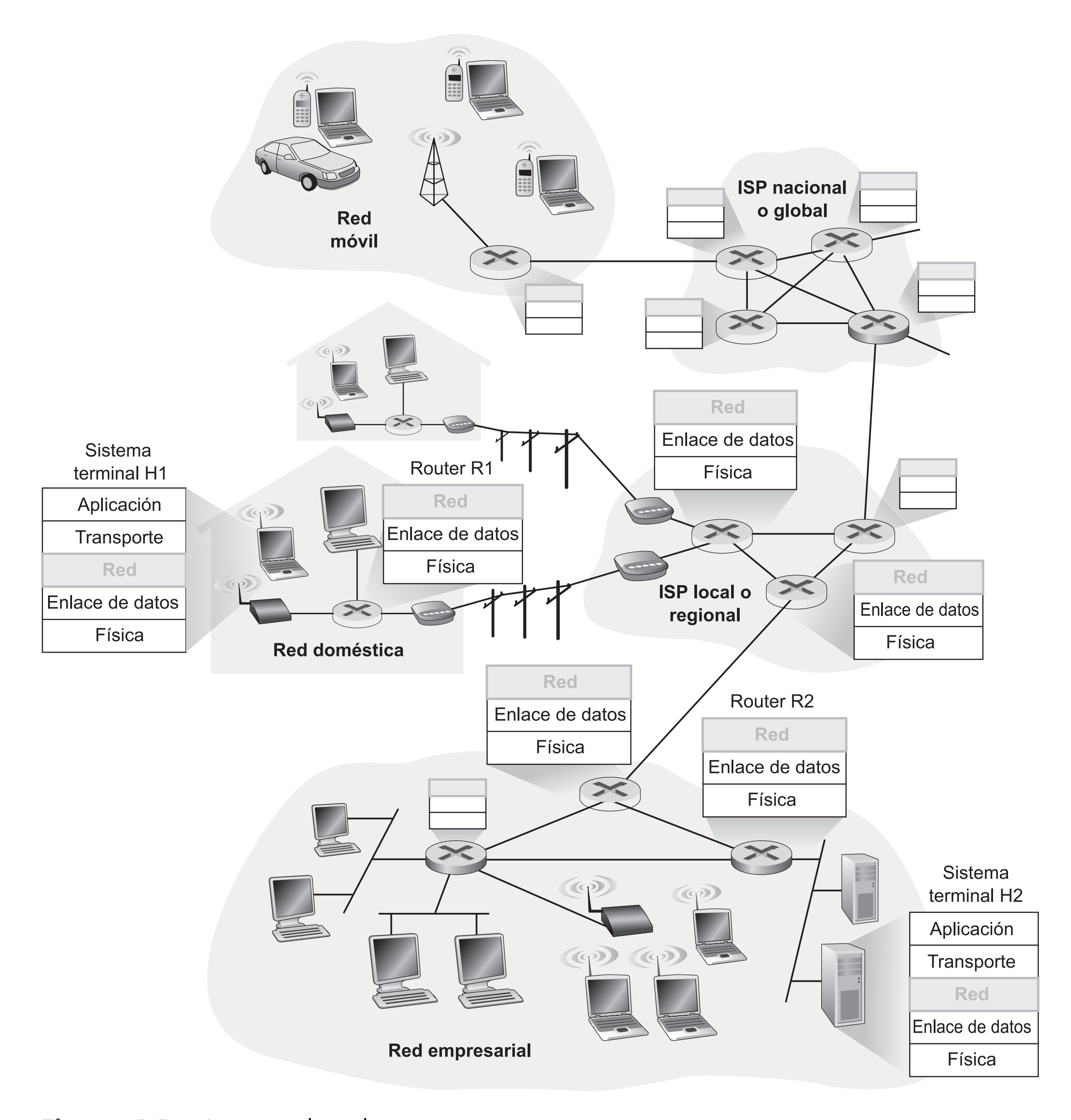


Figura 4.1 • La capa de red.

En la Sección 4.3, miraremos dentro de un router y examinaremos cómo se reenvía realmente un paquete desde un enlace de entrada de un router a uno de sus enlaces de salida.

- *Enrutamiento (routing)*. La capa de red tiene que determinar la ruta o camino que deben seguir los paquetes a medida que fluyen de un emisor a un receptor. Los algoritmos que calculan estas rutas se conocen como **algoritmos de enrutamiento**. Un algoritmo de enrutamiento debe determinar, por ejemplo, la ruta por la que fluirán los paquetes para ir de H1 a H2.

A menudo, los autores que hablan acerca de la capa de red emplean de forma indistinta los términos *reenvío* y *enrutamiento.* Sin embargo, en este libro utilizaremos estos términos de manera más precisa. El *reenvío* hace referencia a la acción local que realiza un router al transferir un paquete desde una interfaz de un enlace de entrada a una interfaz del enlace de salida apropiada. El *enrutamiento* hace referencia al proceso que realiza la red en conjunto para determinar las rutas terminal a terminal que los paquetes siguen desde el origen al destino. Veamos una analogía. Recuerde el viaje desde Pensilvania a Florida que realizó nuestro viajero de la Sección 1.3.2. Durante ese viaje, nuestro conductor atravesó muchas intersecciones en su camino a Florida. Podemos pensar en el reenvío como en el proceso de atravesar una intersección: un coche entra en una intersección viniendo por una carretera y determina qué otra carretera tomar para salir de la intersección. Podemos pensar en el enrutamiento como en el proceso de planificación del viaje desde Pensilvania hasta Florida: antes de iniciar el viaje, el conductor consulta un mapa y elige uno de los muchos posibles caminos, estando cada uno de ellos definido por una serie de tramos de carretera que se conectan en las intersecciones.

Todo router tiene una **tabla de reenvío**. Un router reenvía un paquete examinando el valor de un campo de la cabecera del paquete entrante y utilizando después ese valor para indexarlo dentro de la tabla de reenvío del router. El resultado de la tabla de reenvío indica a cuál de las interfaces del enlace de salida del router será reenviado el paquete. Dependiendo del protocolo de la capa de red, este valor de la cabecera del paquete podría ser la dirección de destino del paquete o una indicación de la conexión a la que pertenece el paquete. La Figura 4.2 proporciona un ejemplo. En esta figura, un paquete con un valor de campo de cabecera de 0111 llega a un router. El router busca en su tabla de reenvío y determina que la interfaz del enlace de salida para este paquete es la interfaz 2. Después, el router reenvía internamente el paquete a la interfaz 2. En la Sección 4.3 veremos cómo el router hace esto y examinaremos la función de reenvío con más detalle.

Por el momento, vamos a ver cómo están configuradas las tablas de reenvío en los routers. Esta cuestión es crucial, ya que expone la importante relación existente entre el enrutamiento y el reenvío. Como se muestra en la Figura 4.2, el algoritmo de enrutamiento determina los valores que se introducen en las tablas de reenvío de los routers. El algoritmo de enrutamiento puede estar centralizado (por ejemplo, con un algoritmo que se ejecute en un sitio central y que descargue la información de enrutamiento en cada router) o descentralizado (esto es, con un componente del algoritmo de enrutamiento distribuido ejecutándose en cada router). En cualquier caso, un router recibe mensajes del protocolo de enrutamiento que utiliza para configurar su tabla de reenvío. La diferencia en los propósitos de las funciones de reenvío y de enrutamiento puede ilustrarse bastante claramente considerando el caso hipotético (y nada realista, pero técnicamente factible) de una red en la que todas las tablas de reenvío fueran configuradas directamente por operadores de red humanos que estuvieran físicamente presentes en los routers. En este caso, ¡*no* se necesitarían protocolos de enrutamiento! Por supuesto, los operadores humanos tendrían que interactuar entre sí para garantizar que las tablas de reenvío estuvieran configuradas de tal forma que los paquetes llegaran a sus destinos. Probablemente también, si esta configuración la hicieran personas sería más propensa a errores y mucho más lenta en responder a los cambios en la topología de la red que un protocolo de enrutamiento. Por tanto, tenemos suerte de que todas las redes dispongan tanto de la función de reenvío como de la de enrutamiento.

Continuando con las cuestiones terminológicas, merece la pena comentar que hay otros dos términos que se emplean indistintamente, pero que nosotros emplearemos con cuidado.

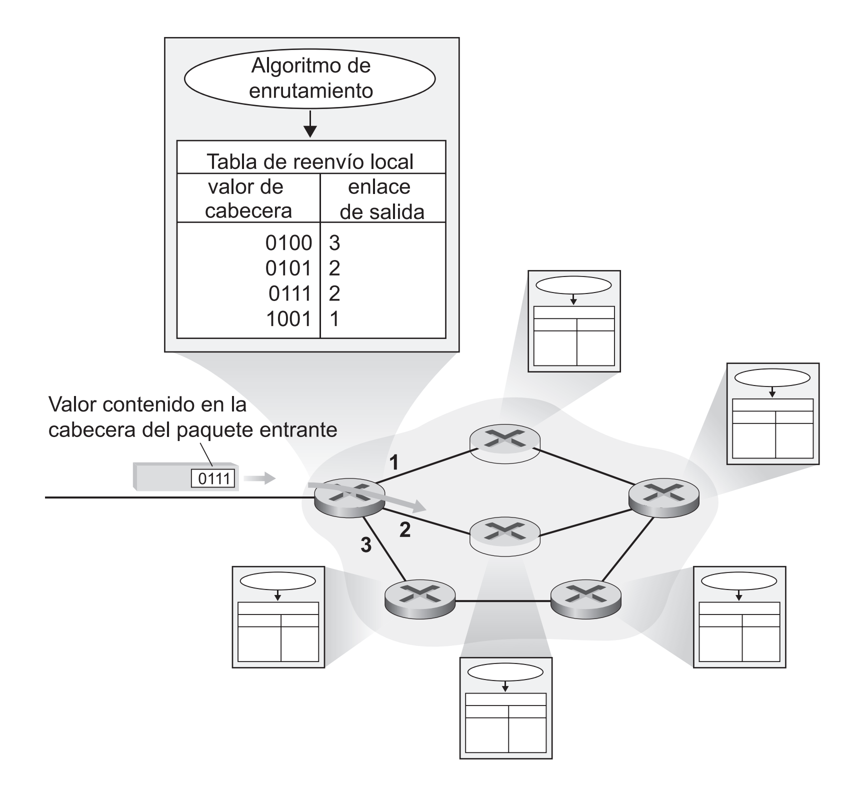


Figura 4.2 • Los algoritmos de enrutamiento determinan los valores almacenados en la tablas de reenvío.

Reservaremos el término *conmutador de paquetes* para referirnos a un dispositivo de conmutación de paquetes general que transfiere un paquete desde la interfaz del enlace de entrada a la interfaz del enlace de salida, de acuerdo con el valor almacenado en un campo de la cabecera del paquete. Algunos conmutadores de paquetes, denominados **conmutadores de la capa de enlace** o **switches** (que se examinan en el Capítulo 5), basan su decisión de reenvío en el valor almacenado en el campo de la capa de enlace. Otros dispositivos de conmutación de paquetes, conocidos como **routers**, basan su decisión de reenvío en el valor almacenado en el campo de la capa de red (con el fin de apreciar esta importante distinción, le invitamos a que repase la Sección 1.5.2, en la que se abordaron los datagramas de la capa de red y las tramas de la capa de enlace y sus relaciones). Puesto que este capítulo está dedicado a la capa de red, utilizaremos el término *router* en lugar de *dispositivo de conmutación de paquetes*. Emplearemos también el término *router* cuando hablemos, en breve, de los conmutadores de paquetes en las redes de circuitos virtuales.

Configuración de la conexión

Acabamos de decir que la capa de red realiza dos importantes funciones: reenvío y enrutamiento. Pero enseguida vamos a ver que en algunas redes de computadoras se lleva a cabo una tercera función de la capa de red importante: la **configuración de la conexión**. Recuerde que hemos visto que en TCP era necesario llevar a cabo un proceso de acuerdo en tres fases antes de que los datos pudieran transferirse del emisor al receptor. Este proceso

permitía al emisor y al receptor configurar la información de estado necesaria (por ejemplo, el número de secuencia y el tamaño de la ventana de control de flujo). De forma análoga, algunas arquitecturas de la capa de red (como por ejemplo ATM y frame-relay, pero no Internet), requieren que los routers a lo largo de la ruta seleccionada desde el origen al destino negocien entre sí para configurar el estado, antes de que puedan comenzar a fluir los paquetes de datos de la capa de red correspondientes a una determinada conexión entre el origen y el destino. En la capa de red, este proceso se conoce como *configuración de la conexión,* la cual estudiaremos en la Sección 4.2.

4.1.2 Modelos de servicio de red

Antes de profundizar en la capa de red, consideremos desde un punto de vista amplio los distintos tipos de servicio que puede ofrecer esta capa. Cuando la capa de transporte de un host emisor transmite un paquete a la red (es decir, lo pasa a la capa de red del host emisor), ¿puede la capa de transporte contar con la capa de red para entregar el paquete al destino? Cuando se envían varios paquetes, ¿se entregan a la capa de transporte del host receptor en el orden en que fueron enviados? ¿El intervalo de tiempo entre el envío de dos transmisiones de paquetes secuenciales será el mismo que el intervalo entre sus respectivas recepciones? ¿Realimentará la red información acerca de la congestión de la misma? ¿Cuál es la visión abstracta (las propiedades) del canal que conecta la capa de transporte en los hosts emisor y receptor? Las respuestas a estas y otras preguntas están determinadas por el modelo de servicio proporcionado por la capa de red. El **modelo de servicio de red** define las características del transporte terminal a terminal de los paquetes entre los sistemas terminales emisor y receptor.

Consideremos ahora algunos de los posibles servicios que podría proporcionar la capa de red. En el host emisor, cuando la capa de transporte pasa un paquete a la capa de red, entre los servicios específicos que la capa de red podría proporcionar se incluyen:

- *Entrega garantizada*. Este servicio garantiza que el paquete terminará por llegar a su destino.
- *Entrega garantizada con retardo limitado*. Este servicio no sólo garantiza la entrega del paquete, sino que dicha entrega tendrá un límite de retardo especificado de host a host (por ejemplo, de 100 milisegundos).

Además, a un *flujo de paquetes* entre un origen y un destino dados podrían ofrecérsele los siguientes servicios:

- *Entrega de los paquetes en orden*. Este servicio garantiza que los paquetes llegan al destino en el orden en que fueron enviados.
- *Ancho de banda mínimo garantizado*. Este servicio de la capa de red emula el comportamiento de un enlace de transmisión con una velocidad de bit específica (por ejemplo, de 1 Mbps) entre los hosts emisor y receptor (incluso aunque la ruta terminal a terminal real pueda atravesar varios enlaces físicos). Mientras que el host emisor transmita los bits (como parte de los paquetes) a una velocidad inferior a la velocidad de bit especificada, no se perderá ningún paquete y todos los paquetes llegarán dentro de un intervalo de retardo host a host pre-especificado (por ejemplo, en 40 milisegundos.)
- *Fluctuación máxima garantizada*. Este servicio garantiza que el intervalo de tiempo transcurrido entre la transmisión de dos paquetes sucesivos en el emisor es igual al intervalo de

tiempo que transcurre entre su respectiva recepción en el destino (es decir, que la separación entre paquetes no variará en una cantidad mayor que un cierto valor especificado).

- *Servicios de seguridad*. Utilizando una clave secreta de sesión sólo conocida por un host de origen y un host de destino, la capa de red del host de origen puede cifrar la carga útil de todos los datagramas que están siendo enviados al host de destino. La capa de red en el host de destino será entonces responsable de descifrar la carga útil. Con un servicio así, se proporcionará confidencialidad a todos los segmentos de la capa de transporte (TCP y UDP) entre los hosts de origen y de destino. Además de la confidencialidad, la capa de red podría ofrecer servicios de integridad de los datos y de autenticación del origen.

Esta lista sólo es una lista parcial de los servicios que la capa de red podría proporcionar (existen incontables posibles variaciones).

La capa de red de Internet proporciona un único servicio conocido como **servicio de mejor esfuerzo (*best-effort service*)**. De acuerdo con la Tabla 4.1, podría parecer que al decir *servicio de mejor esfuerzo* estamos utilizando un eufemismo por no decir que *no proporciona ningún servicio en absoluto*. Con un servicio de mejor esfuerzo, la temporización relativa entre paquetes no está garantizada, tampoco está garantizado que los paquetes se reciban en el orden que fueron emitidos y tampoco se garantiza la entrega de los paquetes transmitidos. Por tanto, teniendo en cuenta esta definición, una red que *no* entregara los paquetes al destino satisfaría la definición de servicio de entrega de mejor esfuerzo. Sin embargo, como veremos enseguida, existen algunas razones válidas para usar tal modelo minimalista de servicio de la capa de red. En el Capítulo 7 veremos algunos modelos de servicio de Internet adicionales, que todavía están evolucionando.

Otras arquitecturas de red han definido e implementado modelos de servicio que van más allá que el servicio de mejor esfuerzo de Internet. Por ejemplo, la arquitectura de red ATM [MFA Forum 2009, Black 1995] proporciona varios modelos de servicio, lo que significa que diferentes conexiones pueden ofrecer distintas clases de servicio dentro de la misma red. Una exposición acerca de cómo una red ATM proporciona tales servicios queda fuera del ámbito de este libro: nuestro fin aquí es únicamente destacar que existen alternativas al modelo de mejor esfuerzo de Internet. Dos de los modelos de servicio de ATM más importantes son el el servicio CBR (*Constant Bit Rate*, tasa de bit constante) y el servicio ABR (*Available Bit Rate*, tasa de bit disponible):

Arquitectura de red	Modelo de servicio	Garantía de ancho de banda	Garantía sin pérdidas	Orden	Temporización	Indicación de congestión
Internet	Mejor esfuerzo	Ninguna	Ninguna	Posible cualquier orden	No se mantiene	Ninguna
ATM	CBR	Velocidad constante garantizada	Sí	En orden	Se mantiene	No se produce congestión
ATM	ABR	Mínimo garantizado	Ninguna	En orden	No se mantiene	Sí

Tabla 4.1 • Modelos de servicio de Internet, y CBR y ABR de redes ATM.

- ***Servicio CBR de las redes ATM.*** Este fue el primer modelo de servicio de las redes ATM en ser estandarizado, lo que refleja el temprano interés de las compañías telefónicas en la tecnología ATM y la adecuación del servicio CBR para el transporte de tráfico de audio y vídeo en tiempo real con una tasa de bit constante. El objetivo del servicio CBR es conceptualmente simple: proporcionar un flujo de paquetes (conocido como celdas en la terminología ATM) mediante un conducto virtual cuyas propiedades son las mismas que si existiera un enlace de transmisión de ancho de banda fijo dedicado entre los hosts emisor y receptor. Con el servicio CBR, un flujo de celdas ATM se transporta a través de la red de tal forma que se garantiza que el retardo terminal a terminal de una celda, la variabilidad del retardo terminal a terminal de una celda (es decir, el *jitter* o fluctuación entre celdas) y la fracción de celdas que se pierden o que se entregan tarde sean todos ellos menores que una serie de valores previamente especificados. El host emisor y la red ATM acuerdan estos valores cuando la conexión CBR se establece por primera vez.
- ***Servicio ABR de las redes ATM.*** Dado que Internet ofrece lo que se denomina un servicio de mejor esfuerzo, el servicio ABR de ATM puede caracterizarse mejor como un servicio ligeramente superior al de mejor esfuerzo. Como con el modelo de servicio de Internet, las celdas se pueden perder con un servicio ABR. Sin embargo, a diferencia de Internet, las celdas no se pueden reordenar (aunque pueden perderse) y está garantizada la velocidad mínima de transmisión de celda (MCR, *Minimum Cell transmission Rate*) de una conexión utilizando el servicio ABR. Si la red tiene los suficientes recursos libres en un instante determinado, un emisor también puede ser capaz de enviar con éxito celdas a una velocidad mayor que la mínima (MCR). Adicionalmente, como hemos visto en la Sección 3.6, el servicio ABR de las redes ATM puede proporcionar una realimentación al emisor (en términos de bit de notificación de congestión, o una velocidad explícita a la que enviar) que controla el modo en que el emisor ajusta su velocidad entre la MCR y un cierto valor máximo de velocidad de celda.

4.2 Redes de circuitos virtuales y de datagramas

Recuerde del Capítulo 3 que la capa de transporte puede ofrecer a las aplicaciones un servicio sin conexión o un servicio orientado a la conexión. Por ejemplo, la capa de transporte de Internet proporciona a cada aplicación la posibilidad de elegir entre dos servicios: UDP, un servicio sin conexión; o TCP, un servicio orientado a la conexión. De forma similar, una capa de red también puede proporcionar un servicio sin conexión o un servicio con conexión. Estos servicios de la capa de red con y sin conexión son paralelos en muchos sentidos a los servicios de la capa de transporte orientados a la conexión y sin conexión. Por ejemplo, un servicio de la capa de red orientado a la conexión comienza con un proceso de acuerdo entre los hosts de origen y de destino; y un servicio de la capa de red sin conexión no realiza ninguna tarea preliminar de acuerdo.

Aunque los servicios con y sin conexión de la capa de red presentan algunos paralelismos con los correspondientes servicios de la capa de transporte, también presentan diferencias importantes:

- En la capa de red, estos servicios son servicios host a host proporcionados por la capa de red a la capa de transporte. En la capa de transporte, estos servicios son servicios proceso a proceso proporcionados por la capa de transporte a la capa de aplicación.

- En las principales arquitecturas de redes de computadoras utilizadas hasta la fecha (Internet, ATM, frame relay, etc.), la capa de red proporciona bien un servicio sin conexión host a host o un servicio orientado a la conexión host a host, pero no ambos. Las redes de computadoras que sólo proporcionan un servicio de conexión en la capa de red se conocen como **redes de circuitos virtuales** (**VC**); las redes que sólo proporcionan un servicio sin conexión en la capa de red se denominan **redes de datagramas**.
- Las implementaciones del servicio orientado a la conexión en la capa de transporte y el servicio con conexión de la capa de red son fundamentalmente distintos. Sabemos del capítulo anterior que el servicio de la capa de transporte orientado a la conexión se implementa en la frontera de la red en los sistemas terminales; enseguida veremos que el servicio de conexión de la capa de red se implementa en los routers del núcleo de la red, así como en los sistemas terminales.

Las redes de circuitos virtuales y de datagramas son dos clases fundamentales de redes de computadoras. Utilizan información muy diferente a la hora de tomar decisiones de reenvío. Veamos a continuación sus implementaciones.

4.2.1 Redes de circuitos virtuales

Hemos estudiado que Internet es una red de datagramas. Sin embargo, muchas arquitecturas de red alternativas (incluyendo las de ATM y frame relay) son redes de circuitos virtuales y, por tanto, utilizan conexiones en la capa de red. Estas conexiones de la capa de red se denominan **circuitos virtuales (VC)**. Veamos ahora cómo se puede implementar un servicio VC en una red de computadoras.

Un circuito virtual consta de (1) una ruta (es decir, una serie de enlaces y routers) entre los hosts de origen y de destino, (2) números de VC, un número para cada enlace a lo largo de la ruta y (3) entradas en la tabla de reenvío de cada router existente a lo largo de la ruta. Un paquete que pertenece a un circuito virtual transportará un número de VC en su cabecera. Dado que un circuito virtual puede tener un número de VC diferente en cada enlace, cada router interviniente tiene que sustituir el número de VC de cada paquete que le atraviesa por un nuevo número de VC. Este nuevo número de VC se obtiene de la tabla de reenvío.

Para ilustrar este concepto, considere la red mostrada en la Figura 4.3. Los números que hay junto a los enlaces de R1 en esta figura son los números de interfaz de enlace. Suponga que el host A solicita a la red que establezca un VC entre él mismo y el host B. Suponga también que la red elige el camino A-R1-R2-B y asigna los números de VC 12, 22 y 32 a los

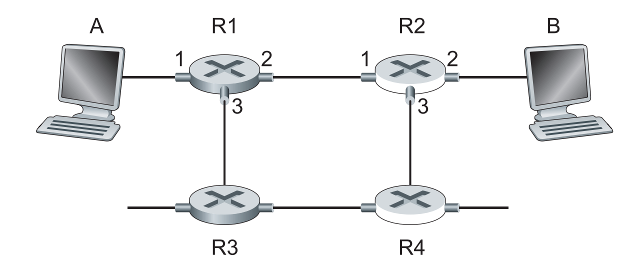


Figura 4.3 • Una red simple de circuitos virtuales.

tres enlaces de ese camino para este circuito virtual. En este caso, cuando un paquete en este circuito virtual abandona el host A, el valor almacenado en el campo número de VC de la cabecera del paquete es 12; cuando sale de R1, el valor es 22; y cuando sale de R2, es 32.

¿Cómo determina el router el número de VC de sustitución para un paquete que le atraviesa? En una red de circuitos virtuales, la tabla de reenvío de cada router incluye la traducción del número de VC; por ejemplo, la tabla de reenvío de R1 sería similar a la siguiente:

Interfaz de entrada	Nº de VC de entrada	Interfaz de salida	Nº de VC salida
1	12	2	22
2	63	1	18
3	7	2	17
1	97	3	87
...	...	...	...

Cuando se configura un número de VC nuevo en un router, se añade una entrada a la tabla de reenvío. De forma similar, cuando un VC termina, las entradas apropiadas de cada tabla a lo largo de la ruta se eliminan.

Es posible que se esté preguntando por qué un paquete no mantiene el mismo número de VC en cada uno de los enlaces de su ruta. La razón es doble. En primer lugar, reemplazar el número en cada enlace reduce la longitud del campo número de VC de la cabecera del paquete. La segunda razón, y más importante, es que la configuración del VC se simplifica considerablemente permitiendo un número de VC diferente en cada enlace a lo largo de la ruta del circuito virtual. Específicamente, con múltiples números de VC, cada enlace de la ruta puede elegir un número de VC, independientemente de los números de VC elegidos en los restantes enlaces a lo largo de la ruta. Si se precisara un número de VC común para todos los enlaces de la ruta, los routers tendrían que intercambiar y procesar un número sustancial de mensajes para acordar un número de VC común (por ejemplo, uno que no estuviera siendo utilizado por cualquier otro VC existente en esos routers) que utilizar en una conexión.

En una red de circuitos virtuales, los routers de la red tienen que mantener **información de estado de la conexión** para las conexiones activas. Específicamente, cada vez que se establece una conexión nueva en un router, tiene que añadirse una nueva entrada de conexión a la tabla de reenvío del router; y cada vez que una conexión se libera, la entrada debe borrarse de la tabla. Observe que incluso si no existe un mecanismo de traducción de los números de VC, seguirá siendo necesario mantener información del estado de la conexión que permita asociar los números de VC con los números de las interfaces de salida. El problema de si un router mantiene o no la información del estado de la conexión para cada conexión activa es crucial, y volveremos repetidamente sobre esta cuestión a lo largo del libro.

En un circuito virtual existen tres fases identificables:

- *Configuración del VC.* Durante la fase de configuración, la capa de transporte del emisor contacta con la capa de red, especifica la dirección del receptor y espera a que la red configure el circuito virtual. La capa de red determina la ruta entre el emisor y el receptor; es decir, la serie de enlaces y routers a través de los que todos los paquetes del VC tendrán que viajar. La capa de red también determina el número de VC para cada enlace de la

ruta. Por último, la capa de red añade una entrada a la tabla de reenvío de cada router existente a lo largo de la ruta. Durante la configuración del VC, la capa de red también puede reservar recursos (por ejemplo, ancho de banda) a lo largo de la ruta del VC.

- *Transferencia de datos.* Como se muestra en la Figura 4.4, una vez que se ha establecido el circuito virtual, los paquetes pueden comenzar a fluir a lo largo del mismo.
- *Terminación del VC.* Esta fase se inicia cuando el emisor (o el receptor) informa a la capa de red de su deseo de terminar el circuito virtual. Normalmente, la capa de red informará al sistema terminal del otro lado de la red de la terminación de la llamada y actualizará las tablas de reenvío de cada uno de los routers de la ruta, para indicar que ese circuito virtual ya no existe.

Existe una sutil pero importante distinción entre la configuración del VC en la capa de red y la configuración de la conexión en la capa de transporte (por ejemplo, el proceso de acuerdo en tres fases de TCP que hemos estudiado en el Capítulo 3). La configuración de la conexión en la capa de transporte sólo implica a los dos sistemas terminales. Durante la configuración de la conexión de la capa de transporte, los dos sistemas terminales solos determinan los parámetros (por ejemplo, el número de secuencia inicial y el tamaño de la ventana de control de flujo) de su conexión de la capa de transporte. Aunque los dos sistemas terminales son conscientes de la conexión de la capa de transporte, los routers de la red la ignoran por completo. Por el contrario, en la capa de red de un circuito virtual *los routers existentes a lo largo de la ruta entre los dos sistemas terminales están implicados en la configuración del VC y cada router es completamente consciente de todos los VC que pasan a través de él.*

Los mensajes que los sistemas terminales envían a la red para iniciar o terminar un VC y los mensajes pasados entre los routers para configurar el VC (es decir, para modificar el estado de conexión en las tablas de los routers) se conocen como **mensajes de señalización** y los protocolos empleados para intercambiar estos mensajes a menudo se denominan **protocolos de señalización**. En la Figura 4.4 se muestra gráficamente la configuración del VC. En este libro no vamos a estudiar los protocolos de señalización de los circuitos virtuales; puede consultar [Black 1997] para ver una exposición general sobre la señalización en las redes orientadas a la conexión y [ITU-T Q.2931 1994] para ver la especificación del protocolo de señalización Q.2931 de ATM.

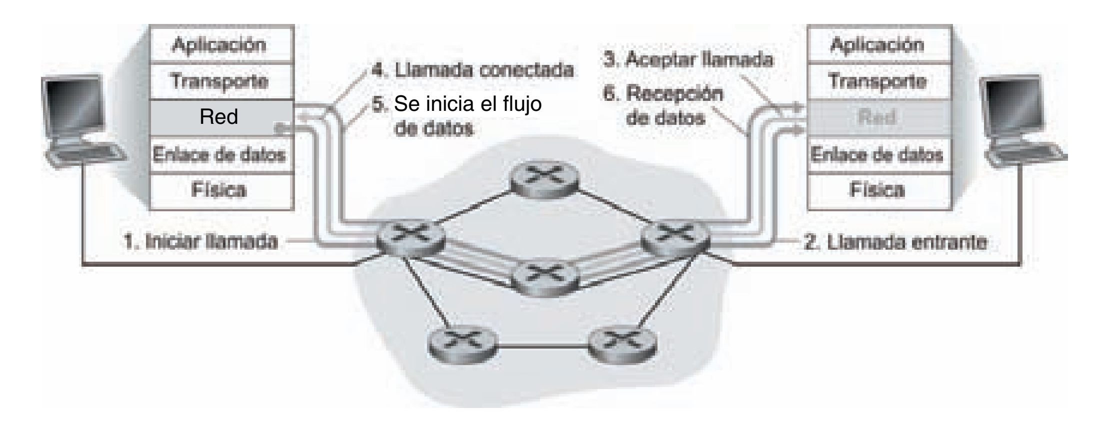


Figura 4.4 • Configuración del circuito virtual.

4.2.2 Redes de datagramas

En una **red de datagramas**, cada vez que un sistema terminal desea enviar un paquete marca el paquete con la dirección del sistema terminal de destino y luego introduce el paquete en la red. Como se muestra en la Figura 4.5, esto se hace sin configurar ningún circuito virtual. Los routers de una red de datagramas no mantienen ninguna información de estado acerca de los circuitos virtuales (¡porque no existe ningún circuito virtual!).

Cuando un paquete se transmite desde un origen a un destino pasa a través de una serie de routers. Cada uno de estos routers utiliza la dirección de destino del paquete para reenviar dicho paquete. Específicamente, cada router tiene una tabla de reenvío que asigna direcciones de destino a interfaces de enlace; cuando un paquete llega a un router, éste utiliza la dirección de destino del paquete para buscar la interfaz del enlace de salida apropiado en la tabla de reenvío. Después, el router reenvía intencionadamente el paquete a esa interfaz del enlace de salida.

Profundicemos algo más en la operación de búsqueda mediante un ejemplo concreto. Suponga que todas las direcciones de destino tienen una longitud de 32 bits (que es la longitud de la dirección de destino de un datagrama IP). Una implementación por fuerza bruta de la tabla de reenvío tendría una entrada para cada una de las posibles direcciones de destino. Dado que existen más de 4.000 millones de posibles direcciones, esta opción está totalmente fuera de lugar (requeriría una tabla de reenvío enormemente grande).

Supongamos ahora que nuestro router tiene cuatro enlaces, numerados de 0 a 3, y que los paquetes deben ser reenviados a las interfaces de enlace como sigue:

Rango de direcciones de destino	Interfaz de enlace
11001000 00010111 00010000 00000000 hasta 11001000 00010111 00010111 11111111	0
11001000 00010111 00011000 00000000 hasta 11001000 00010111 00011000 11111111	1
11001000 00010111 00011001 00000000 hasta 11001000 00010111 00011111 11111111	2
en otro caso	3

Evidentemente, para este ejemplo no es necesario disponer de 4.000 millones de entradas en la tabla de reenvío del router. Podríamos, por ejemplo, tener la siguiente tabla de reenvío con sólo cuatro entradas:

Coincidencia de prefijo	Interfaz de enlace
11001000 00010111 00010	0
11001000 00010111 00011000	1
11001000 00010111 00011	2
en otro caso	3

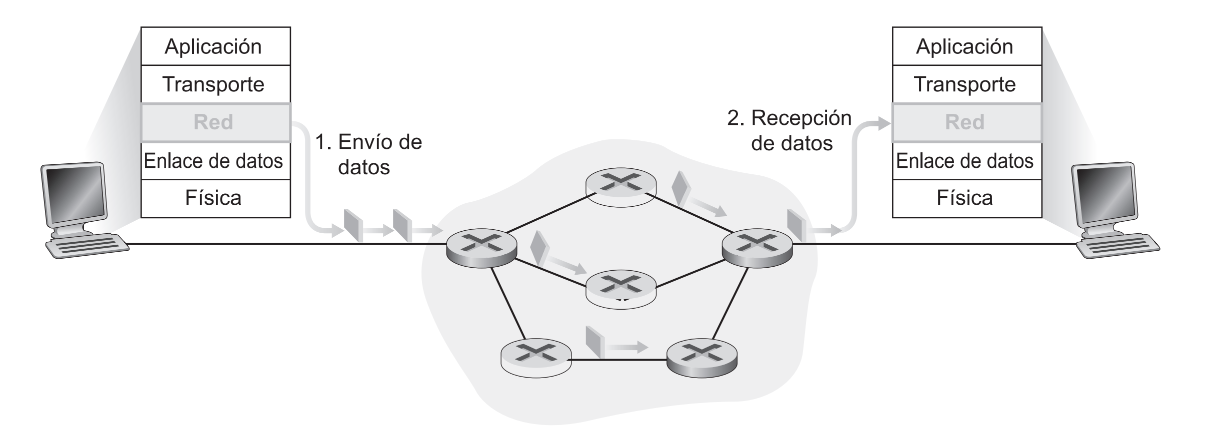


Figura 4.5 • Red de datagramas.

Con este tipo de tabla de reenvío, el router busca la coincidencia de un **prefijo** de la dirección de destino del paquete con las entradas de la tabla; si existe una coincidencia, el router reenvía el paquete a un enlace asociado con esa coincidencia. Por ejemplo, suponga que la dirección de destino del paquete es 11001000 00010111 00010110 10100001; puesto que el prefijo de 21 bits de esta dirección coincide con la primera entrada de la tabla, el router reenvía el paquete a la interfaz de enlace 0. Si un prefijo no se corresponde con ninguna de las tres primeras entradas, entonces el router reenvía el paquete a la interfaz 3. Aunque este método puede parecer bastante simple, esconde una importante sutileza. Es posible que se haya dado cuenta de que la dirección de destino puede corresponderse con más de una entrada. Por ejemplo, los primeros 24 bits de la dirección 11001000 00010111 00011000 10101010 coinciden con la segunda entrada de la tabla y los primeros 21 bits con la tercera entrada de la tabla. Cuando existen varias coincidencias, el router aplica la **regla de coincidencia con el prefijo más largo**; es decir, busca la entrada más larga de la tabla con la que exista una coincidencia y reenvía el paquete a la interfaz de enlace asociada con el prefijo más largo. Cuando estudiemos con más detalle, en la Sección 4.4, el direccionamiento de Internet veremos por qué se utiliza esta regla de coincidencia con el prefijo más largo.

Aunque los routers en las redes de datagramas no mantienen información del estado de la conexión, sí mantienen información del estado de reenvío en sus tablas de reenvío. Sin embargo, la escala de tiempo a la que esta información del estado de reenvío cambia es relativamente lenta. De hecho, en una red de datagramas las tablas de reenvío son modificadas por los algoritmos de enrutamiento, que normalmente actualizan una tabla de reenvío en intervalos aproximados de entre uno y cinco minutos. En una red de circuitos virtuales, la tabla de reenvío de un router se modifica cada vez que se configura una nueva conexión a través del router o cuando una conexión existente en el router se termina. Este proceso puede tener lugar con una escala de tiempo del orden de microsegundos en un router troncal de nivel 1.

Puesto que las tablas de reenvío en las redes de datagramas pueden ser modificadas en cualquier instante, una serie de paquetes enviados desde un sistema terminal a otro puede seguir caminos distintos a través de la red y pueden llegar desordenados. [Paxson 1997] y [Jaiswal 2003] presentan estudios interesantes de medición para la reordenación de paquetes y otros fenómenos de Internet.

4.2.3 Orígenes de las redes de circuitos virtuales y de datagramas

La evolución de las redes de datagramas y de circuitos virtuales refleja sus orígenes. La idea de un circuito virtual como principio de organización central tiene sus raíces en el mundo de la telefonía, que utiliza circuitos reales. Al mantenerse la configuración y el estado de las llamadas en los routers de la red, una red de circuitos virtuales es indiscutiblemente más compleja que una red de datagramas (aunque puede consultar [Molinero-Fernández 2002] para ver una interesante comparación de la complejidad de las redes de circuitos y las redes de conmutación de paquetes). Esto también se debe a su herencia del campo de la telefonía. Las redes telefónicas, por necesidad, mantenían la complejidad dentro de la red, ya que estaban conectadas a dispositivos terminales no inteligentes, como los teléfonos de disco. Para aquellos que sean demasiado jóvenes como para saberlo, un teléfono de disco es un teléfono analógico sin teclas, y con sólo un dial.

Por otro lado, Internet, como red de datagramas, creció a partir de la necesidad de conectar computadoras entre sí. Con dispositivos terminales sofisticados, los arquitectos de Internet prefirieron hacer que el modelo de servicio de la capa de red fuera lo más simple posible. Como ya hemos visto en los Capítulos 2 y 3, las funcionalidades adicionales, como por ejemplo la entrega de los datos en orden, la transferencia de datos fiable, el control de congestión y la resolución de nombres DNS, se implementan en una capa más alta en los sistemas terminales. Esto invierte el modelo de las redes de telefonía, lo que tiene algunas consecuencias interesantes:

- Puesto que el modelo de servicio de la capa de red de Internet proporciona unas garantías de servicio mínimas (¡ninguna!), impone unos requisitos mínimos a la capa de red. Esto hace que sea más fácil interconectar redes que utilizan tecnologías de la capa de enlace muy diferentes (por ejemplo, conexión vía satélite, Ethernet, fibra o radio) y que presentan características de pérdidas y velocidades de transmisión muy distintas. Veremos en detalle la interconexión de redes IP en la Sección 4.4.
- Como hemos visto en el Capítulo 2, las aplicaciones tales como el correo electrónico, la Web e incluso un servicio como el de DNS (centralizado en la capa de red) se implementan en hosts (servidores) situados en la frontera de la red. La capacidad de añadir un nuevo servicio simplemente conectando un host a la red y definiendo un nuevo protocolo de la capa de aplicación (como HTTP) ha permitido a las nuevas aplicaciones como la Web implantarse en Internet en un periodo de tiempo extremadamente corto.

Como veremos en el Capítulo 7, existe un debate importante en la comunidad de Internet acerca de cómo debería evolucionar la arquitectura de la capa de red de Internet para dar soporte a los servicios en tiempo real, como por ejemplo multimedia. En [Crowcroft 1995] puede encontrar una interesante comparación de la arquitectura de red de ATM, orientada a circuitos virtuales, y de una arquitectura propuesta para la Internet de siguiente generación.

4.3 El interior de un router

Ahora que ya hemos visto una introducción a las funciones y servicios de la capa de red, vamos a prestar atención a la **función de reenvío** de la capa de red: la transferencia real de paquetes desde los enlaces de entrada de un router a los apropiados enlaces de salida. Ya hemos hablado brevemente de algunos de los problemas de la función de reenvío en la Sec-

ción 4.2; en concreto, del direccionamiento y de la coincidencia con el prefijo más largo. En esta sección veremos arquitecturas de router específicas para la transferencia de paquetes desde los enlaces de entrada a los enlaces de salida. Nuestra exposición va a ser necesariamente breve, ya que se necesitaría un curso completo para cubrir en profundidad el diseño de un router. En consecuencia, haremos un esfuerzo especial en esta sección para proporcionar referencias a textos que abordan este tema con más detalle. Ya hemos comentado que los profesionales e investigadores dedicados al mundo de las redes de computadoras suelen emplear indistintamente los términos *reenvío* y *conmutación,* por lo que nosotros utilizaremos ambos términos en el libro.

En la Figura 4.6 se muestra un esquema general de la arquitectura de un router genérico. En un router, podemos identificar los cuatro componentes siguientes:

- *Puertos de entrada.* El puerto de entrada realiza varias funciones. Lleva a cabo las funciones de la capa física (representadas por el recuadro situado más a la izquierda del puerto de entrada y el recuadro más a la derecha del puerto de salida en la Figura 4.6) consistentes en la terminación de un enlace físico de entrada a un router. Realiza las funciones de la capa de enlace de datos (representadas por los recuadros centrales de los puertos de entrada y de salida) necesarias para interoperar con las funciones de la capa de enlace de datos en el lado remoto del enlace de entrada. También realiza una función de búsqueda y reenvío (el recuadro más a la derecha del puerto de entrada y el recuadro más a la izquierda del puerto de salida) de modo que un paquete reenviado dentro del entramado de conmutación del router emerge en el puerto de salida apropiado. Los paquetes de control (por ejemplo, paquetes que transportan la información del protocolo de enrutamiento) son reenviados desde un puerto de entrada al procesador de enrutamiento. En la práctica, suelen agruparse varios puertos en una única **tarjeta de línea (*line card*)** dentro del router.
- *Entramado de conmutación.* El entramado de conmutación conecta los puertos de entrada del router a sus puertos de salida. Este entramado de conmutación está completamente contenido dentro del router: ¡una red dentro de un router de red!

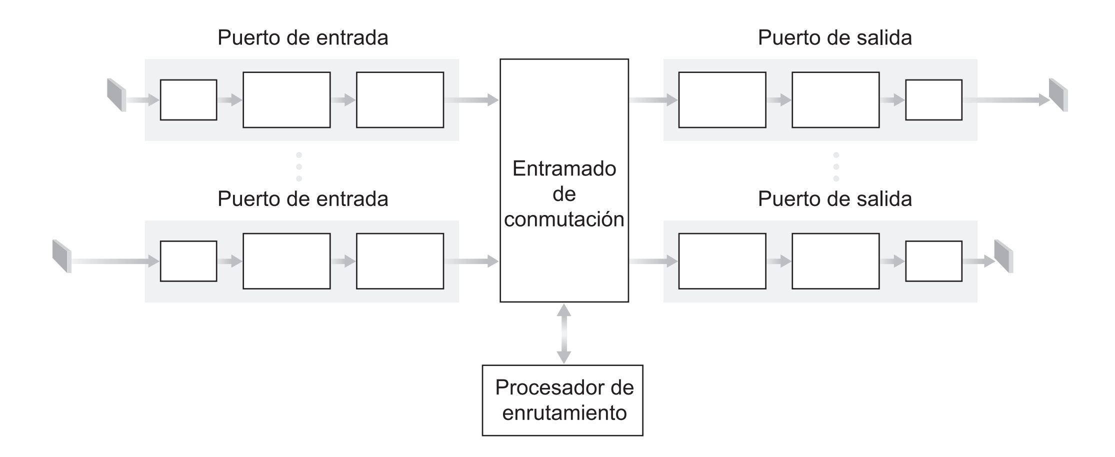


Figura 4.6 • Arquitectura de un router.

- *Puertos de salida.* Un puerto de salida almacena los paquetes que le han sido reenviados a través del entramado de conmutación y los transmite al enlace de salida. Así, el puerto de salida lleva a cabo la función inversa de la capa física y de la capa de enlace de datos que el puerto de entrada. Cuando un enlace es bidireccional (es decir, transporta tráfico en ambas direcciones), un puerto de salida del enlace normalmente estará emparejado con otro puerto de entrada de dicho enlace en la misma tarjeta de línea.
- *Procesador de enrutamiento*. El procesador de enrutamiento ejecuta los protocolos de enrutamiento (por ejemplo, los protocolos que se estudian en la Sección 4.6), mantiene la información de enrutamiento y las tablas de reenvío y realiza funciones de gestión de red (véase el Capítulo 9) dentro del router.

En las siguientes subsecciones veremos los puertos de entrada, el entramado de conmutación y los puertos de salida en más detalle. [Chuang 2005; Keslassy 2003; Chao 2001; Turner 1988; Giacopelli 1990; McKeown 1997a; Partridge 1998] proporcionan información acerca de algunas arquitecturas de router específicas. [McKeown 1997b] proporciona una introducción particularmente fácil de comprender de las arquitecturas de router actuales, utilizando el router 12000 de Cisco como ejemplo. Para concretar, en la siguiente exposición suponemos que la red de computadoras es una red de paquetes y que las decisiones de reenvío están basadas en la dirección de los paquetes (en lugar de en un número de VC de una red de circuitos virtuales). No obstante, los conceptos y técnicas son similares para una red de circuitos virtuales.

4.3.1 Puertos de entrada

En la Figura 4.7 se muestra un esquema detallado de la funcionalidad de un puerto de entrada. Como hemos comentado anteriormente, la función de terminación de línea y el procesamiento de enlace de datos del puerto de entrada implementan las capas física y de enlace de datos asociadas con un enlace individual de entrada al router. El módulo de búsqueda/reenvío del puerto de entrada resulta crucial para la función de reenvío del router. En muchos routers, es aquí donde el router determina el puerto de salida al que será reenviado un paquete entrante a través del entramado de conmutación. La elección del puerto de salida se lleva a cabo utilizando la información almacenada en la tabla de reenvío. Aunque el procesador de enrutamiento calcula la tabla de reenvío, suele almacenarse una copia de la misma en cada puerto de entrada, la cual es actualizada según sea necesario por el procesador de enrutamiento. Con las copias locales de la tabla de reenvío, la decisión de reenvío puede tomarse localmente en cada puerto de entrada sin invocar al procesador de enruta-

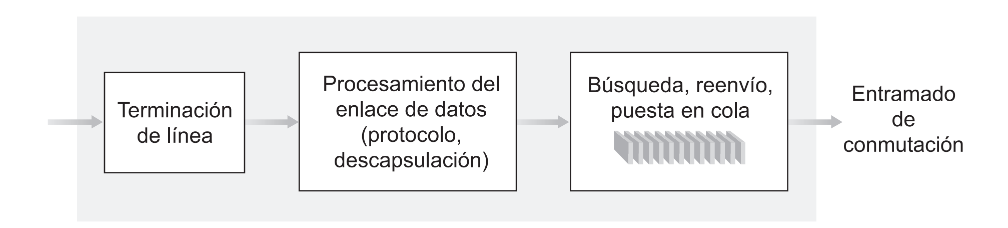


Figura 4.7 • Procesamiento en el puerto de entrada.



HISTORIA

CISCO SYSTEMS: DOMINIO DEL NÚCLEO DE LA RED

En el momento de escribir esto (octubre de 2008), Cisco emplea a más de 65.000 personas. Actualmente, Cisco domina el mercado de los routers de Internet y en los últimos años ha entrado en el mercado de la telefonía por Internet, donde compite con empresas líderes dedicadas a equipos de telefonía como Lucent, Alcatel, Nortel y Siemens. ¿Cómo nació este monstruo de empresa dedicada a las redes? Todo empezó en 1984 en el cuarto de estar de un apartamento en Silicon Valley.

Len Bosak y su esposa Sandy Lerner estaban trabajando en la Universidad de Stanford cuando tuvieron la idea de crear y vender routers Internet a instituciones académicas y de investigación. Sandy Lerner inventó el nombre de Cisco (una abreviatura de San Francisco), y también diseñó el logotipo del puente que identifica a la empresa. Sus oficinas corporativas estaban en su cuarto de estar y financiaron el proyecto con tarjetas de crédito y trabajos de consultoría en horas libres. A finales de 1986, los ingresos de Cisco alcanzaban los 250.000 dólares mensuales. A finales de 1987, Cisco logró atraer inversores de capital riesgo: 2 millones de dólares de Sequoia Capital a cambio de una tercera parte de la compañía. A lo largo de los años siguientes, Cisco continuó creciendo y consiguiendo cada vez una mayor cuota de mercado. Al mismo tiempo, las relaciones entre Bosak/Lerner y la dirección de Cisco comenzaron a tensarse. Cisco empezó a cotizar en bolsa en 1990; ese mismo año, Lerner y Bosak dejaban la compañía.

A lo largo de los años, Cisco se ha expandido más allá del mercado de los routers, y vende servicios y productos de seguridad, inalámbricos y de voz sobre IP. Sin embargo, Cisco se enfrenta a una creciente competencia internacional, que incluye a Huawei, una compañía china de rápido crecimiento dedicada al mercado de las redes. Otras empresas competidoras de Cisco en el campo de los routers y de las redes Ethernet conmutadas son Alcatel-Lucent y Juniper.

miento centralizado. Tales reenvíos *descentralizados* evitan que se formen cuellos de botella en el procesamiento de reenvíos en un único punto dentro del router.

En los routers con capacidades de procesamiento limitadas en el puerto de entrada, éste puede simplemente reenviar el paquete al procesador de enrutamiento centralizado, el cual entonces hará una búsqueda en la tabla de reenvío y transmite el paquete al puerto de salida apropiado. Ésta es la técnica que se aplica cuando una estación de trabajo o un servidor se emplea como un router; en este caso, el procesador de enrutamiento es realmente la CPU de la estación de trabajo y el puerto de entrada es una tarjeta de interfaz de red (por ejemplo, una tarjeta Ethernet).

Disponiendo de una tabla de reenvío, la operación de búsqueda resulta conceptualmente simple: basta con recorrer la tabla de reenvío hasta localizar la coincidencia con el prefijo más largo, como se ha descrito en la Sección 4.2.2. Sin embargo, en la práctica, las cosas no son tan sencillas. Quizá el factor más importante que lo complica es que los routers troncales tienen que operar a velocidades altas, realizando millones de búsquedas por segundo. De hecho, es deseable que el procesamiento en el puerto de entrada pueda realizarse a la **velocidad de línea**, es decir, de manera que una búsqueda se lleve a cabo en menos tiempo que el necesario para recibir un paquete a través del puerto de entrada. En este

caso, el procesamiento de entrada de un paquete recibido puede completarse antes de que se concluya la siguiente operación de recepción. Para hacerse una idea de los requisitos de rendimiento de una búsqueda, considere que un enlace OC-48 opera a 2,5 Gbps. Con paquetes cuya longitud es de 256 bytes, esto implica una velocidad de búsqueda de aproximadamente 1 millón de búsquedas por segundo.

Debido a la necesidad actual de operar a velocidades de enlace altas, una búsqueda lineal a través de una tabla de reenvío grande es imposible. Una técnica más razonable consiste en almacenar las entradas de la tabla de reenvío en una estructura de datos en árbol. Cada nivel del árbol puede interpretarse como el correspondiente a un bit de la dirección de destino. Para buscar una dirección, simplemente se comienza por el nodo raíz del árbol. Si el primer bit de la dirección es cero, entonces el subárbol de la izquierda contendrá la entrada de la tabla de reenvío correspondiente a la dirección de destino; en cualquier otro caso, se encontrará en el subárbol de la derecha. A continuación, se recorre el subárbol apropiado utilizando los restantes bits de la dirección (si el siguiente bit es cero, se elegirá el subárbol izquierdo del subárbol inicial; en otro caso, se seleccionará el subárbol derecho del subárbol inicial). De esta forma, es posible buscar la entrada de la tabla de reenvío en N pasos, siendo N el número de bits de la dirección (observe que se trata de una búsqueda binaria en un espacio de direcciones de tamaño 2^N). En [Srinivasan 1999] se describe una mejora de las técnicas de búsqueda binaria y puede encontrar un estudio general de los algoritmos de clasificación de paquetes en [Gupta 2001].

Pero incluso con $N = 32$ pasos (por ejemplo, para una dirección IP de 32 bits), la velocidad en una búsqueda binaria no es lo suficientemente rápida para los requisitos de enrutamiento de una red troncal actual. Por ejemplo, suponiendo un acceso a memoria en cada paso, se podrían realizar menos de un millón de búsquedas de direcciones por segundo con un tiempo de acceso a memoria de 40 ns. Por tanto, ha sido necesario explorar otras técnicas con el fin de incrementar las velocidades de búsqueda. Las **memorias direccionables por contenido (CAM, *Content Addressable Memory*)** permiten acceder a la memoria mediante una dirección IP de 32 bits, devolviendo la memoria el contenido de la correspondiente entrada de la tabla de reenvío en un tiempo prácticamente constante. Los routers de la serie 8500 de Cisco utilizan una CAM de 64k para cada puerto de entrada.

Otra técnica que permite acelerar las búsquedas consiste en mantener las entradas de la tabla de reenvío a las que se ha accedido recientemente en una caché [Feldmeier 1988]. En este caso, el problema se encuentra en el tamaño potencial de la caché. Se han propuesto también estructuras de datos rápidas, que permiten localizar las entradas de la tabla de reenvío en log(N) pasos [Waldvogel 1997], o que comprimen las tablas de reenvío en formas originales [Brodnik 1997]. En [Gupta 1998] se expone una técnica hardware de búsqueda que está optimizada para el caso común en el que las direcciones que tienen que localizarse tienen 24 o menos bits significativos. Para ver un estudio y una taxonomía de los algoritmos de búsqueda de direcciones IP de alta velocidad, consulte [Ruiz-Sánchez 2001].

Una vez que se ha determinado el puerto de salida para un paquete mediante una búsqueda, el paquete puede ser reenviado hacia el entramado de conmutación. Sin embargo, un paquete puede ver temporalmente bloqueada su entrada en el entramado de conmutación (porque los paquetes de otros puertos de entrada pueden estar utilizando en este momento dicho entramado). En ese caso, el paquete bloqueado deberá ponerse en cola en el puerto de entrada y planificarse para cruzar el entramado de conmutación en un momento posterior. En la Sección 4.3.4 veremos más en detalle el proceso de bloqueo, puesta en cola y planificación de los paquetes (tanto en los puertos de entrada como de salida) dentro de un router.

4.3.2 Entramado de conmutación

El entramado de conmutación se encuentra en el corazón de un router. Es a través de este entramado donde los paquetes son realmente conmutados (es decir, reenviados) desde un puerto de entrada a un puerto de salida. La conmutación puede llevarse a cabo de varias formas, como se muestra en la Figura 4.8:

- *Conmutación vía memoria.* Los primeros routers más simples eran computadoras tradicionales en las que la conmutación entre los puertos de entrada y de salida se realizaba bajo el control directo de la CPU (procesador de enrutamiento). Los puertos de entrada y de salida funcionaban como dispositivos de E/S tradicionales en un sistema operativo tradicional. El puerto de entrada al que llegaba un paquete enviaba una señal en primer lugar al procesador de enrutamiento mediante una interrupción. A continuación, el paquete se copiaba desde el puerto entrada en la memoria del procesador. Después, el procesador de enrutamiento extraía la dirección de destino de la cabecera, buscaba en la tabla de reenvío el puerto de salida apropiado y copiaba el paquete en los buffers del puerto de salida. Observe que si el ancho de banda de la memoria es tal que pueden escribirse en, o leerse de, la memoria *B* paquetes por segundo, entonces la tasa de transferencia global de reenvío (la velocidad total a la que los paquetes se transfieren desde los puertos de entrada a los de salida) tiene que ser menor que *B/*2.

  Muchos routers modernos también realizan la conmutación vía memoria. Sin embargo, una diferencia importante con los primeros routers es que los procesadores de las tarjetas

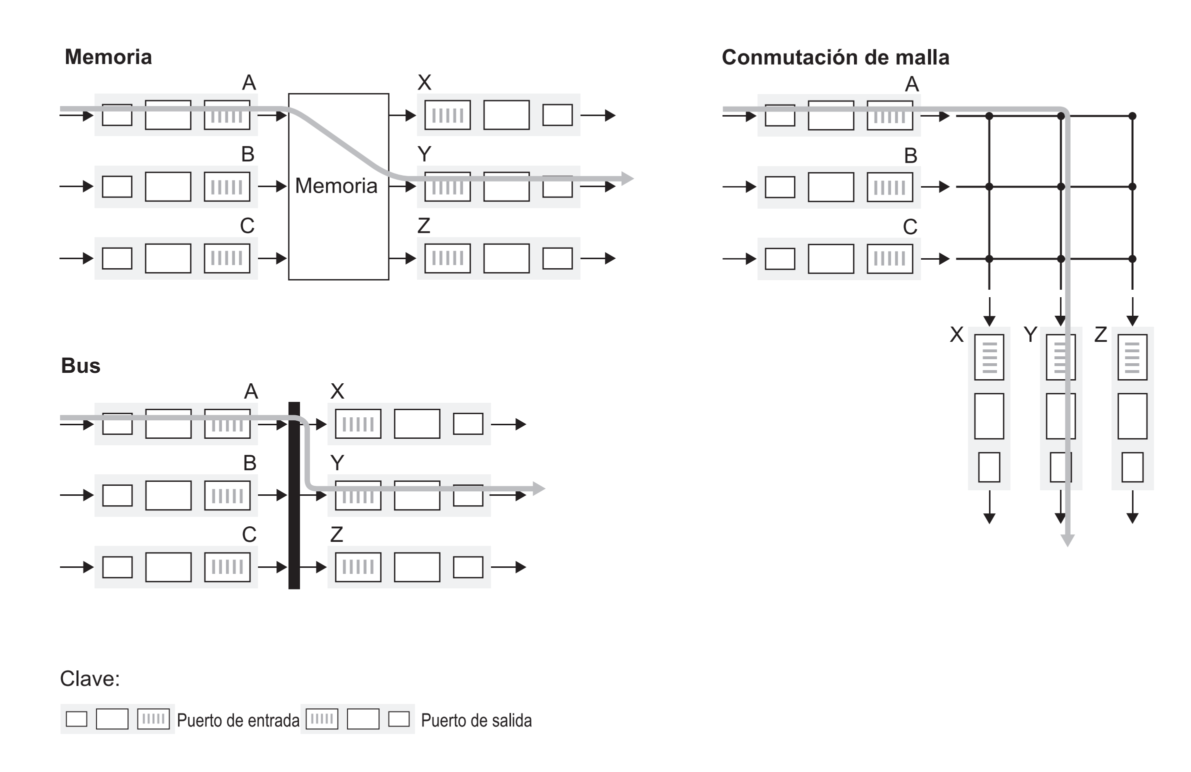


Figura 4.8 • Tres técnicas de conmutación.

de línea de entrada se encargan de realizar la búsqueda de la dirección de destino y de almacenar el paquete en la posición de memoria adecuada. De alguna forma, los routers que conmutan vía memoria se parecen mucho a los multiprocesadores de memoria compartida, encargándose los procesadores de una tarjeta de línea de conmutar los paquetes hacia la memoria del puerto de salida apropiado. Los dispositivos de conmutación (*switches*) de la serie Catalyst 8500 de Cisco [Cisco 8500 2009] reenvían los paquetes mediante una memoria compartida. Puede encontrar un modelo abstracto para el estudio de las propiedades de la conmutación basada en memoria y una comparación con otros métodos de conmutación en [Iyer 2002].

- *Conmutación vía bus.* Con esta técnica, el puerto de entrada transfiere directamente un paquete al puerto de salida a través de un bus compartido sin intervención del procesador de enrutamiento (observe que con la conmutación vía memoria, el paquete también tenía que atravesar el bus del sistema para ir y venir de la memoria). Aunque el procesador de enrutamiento no está implicado en la transferencia del bus, puesto que se trata de un bus compartido, sólo es posible transferir un paquete cada vez a través del bus. Un paquete que llega a un puerto de entrada y se encuentra con que el bus está ocupado con la transferencia de otro paquete queda bloqueado y no se le permite atravesar el entramado de conmutación, poniéndose en cola en el puerto de entrada. Puesto que todos los paquetes tienen que atravesar el único bus, el ancho de banda de conmutación del router está limitado por la velocidad del bus.

  Puesto que con la tecnología actual es posible disponer de anchos de banda de bus de aproximadamente 1 Gbps, la conmutación vía bus suele ser suficiente para routers que operan en redes de acceso y empresariales (como por ejemplo, redes locales y corporativas). La conmutación basada en bus ha sido adoptada en bastantes routers actuales, incluyendo los Cisco 5600 [Cisco Switches 2009], que conmutan los paquetes sobre un bus de tipo backplane de 32 Gbps.

- *Conmutación vía una red de interconexión.* Una forma de soslayar la limitación del ancho de banda de un único bus compartido consiste en emplear una red de interconexión más sofisticada, como las que se han empleado en el pasado para interconectar procesadores en una arquitectura de computadora multiprocesador. Un conmutador de malla (*crossbar switch*) es una red de interconexión que consta de $2n$ buses que conectan n puertos de entrada a n puertos de salida, como se muestra en la Figura 4.8. Un paquete que llega a un puerto de entrada viaja a lo largo del bus horizontal conectado al puerto de entrada hasta intersectar con el bus vertical que le dirige al puerto de salida deseado. Si este bus vertical está libre, el paquete se transfiere al puerto de salida. Si el bus vertical está siendo utilizado para transferir un paquete procedente de otro puerto de entrada a este mismo puerto de salida, el paquete entrante quedará bloqueado y en cola en el puerto de entrada.

  Los entramados de conmutación Delta y Omega también han sido propuestos como una red de interconexión entre los puertos de entrada y de salida. Consulte [Tobagi 1990] para ver un estudio sobre las arquitecturas de los conmutadores. Los conmutadores o switches de la familia 12000 de Cisco [Cisco 12000 2009] utilizan una red de interconexión, proporcionando 60 Gbps en el entramado de conmutación. Una tendencia en el diseño de redes de interconexión [Keshav 1998] es la de fragmentar un paquete IP de longitud variable en celdas de longitud fija, y después etiquetar y conmutar dichas celdas a través de la red de interconexión. Estas celdas son reensambladas después para volver a formar el paquete original en el puerto de salida. Las celdas de longitud fija y las etique-

tas internas pueden simplificar y acelerar considerablemente la conmutación del paquete a través de la red de interconexión.

4.3.3 Puertos de salida

El procesamiento en los puertos de salida, como se muestra en la Figura 4.9, toma los paquetes que hayan sido almacenados en la memoria del puerto de salida y los transmite a través del enlace de salida. El procesamiento del protocolo de enlace de datos y la terminación de línea son funcionalidades de las capas física y de enlace del lado emisor que interactúan con el puerto de entrada en el otro extremo del enlace de salida, como se ha visto en la Sección 4.3.1. La funcionalidad de gestión de la cola y de buffer es necesaria cuando el entramado de conmutación suministra paquetes al puerto de salida a una velocidad mayor que la velocidad del enlace de salida; más adelante veremos la gestión de la cola en los puertos de salida.

4.3.4 ¿Dónde se crean colas?

Si nos fijamos en la funcionalidad de los puertos de entrada y de salida, y en las configuraciones mostradas en la Figura 4.8, es evidente que pueden formarse colas de paquetes en los puertos de entrada *y* en los puertos de salida. Es importante estudiar estas colas con un poco más de detalle, ya que a medida que estas colas crecen, el espacio disponible en el buffer del router terminará agotándose y se producirá una **pérdida de paquetes**. Recuerde que en nuestras explicaciones anteriores hemos dicho que los paquetes se perdían dentro de la red o eran descartados por un router. Por tanto, es en estas colas de los routers donde los paquetes realmente se descartan y se pierden. El lugar en el que se pierde el paquete (en las colas del puerto de entrada o en las colas del puerto de salida) dependerá de la carga de tráfico, de la velocidad relativa del entramado de conmutación y de la velocidad de línea, como veremos enseguida.

Suponga que las velocidades de línea de entrada y las velocidades de línea de salida son todas ellas idénticas y que existen *n* puertos de entrada y *n* puertos de salida. Definimos la **velocidad del entramado de conmutación** como la velocidad a la que dicho entramado puede mover los paquetes desde los puertos de entrada hasta los puertos de salida. Si la velocidad del entramado de conmutación es al menos *n* veces mayor que la velocidad de línea de entrada, entonces nunca habrá colas en los puertos de entrada. Esto es así, porque incluso en el caso peor, en el que las *n* líneas de entrada estén recibiendo paquetes, el conmutador podrá transferir *n* paquetes desde el puerto de entrada al de salida en el tiempo que tarda cada uno de los *n* puertos de entrada en recibir (simultáneamente) un *único* paquete. ¿Pero, qué puede

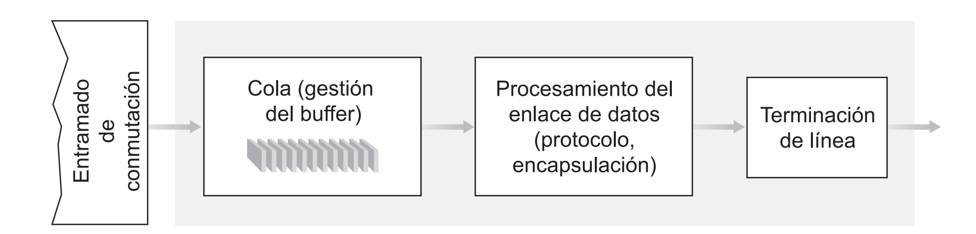


Figura 4.9 • Procesamiento del puerto de salida.

ocurrir en los puertos de salida? Supongamos además que la estructura de conmutación es al menos n veces tan rápida como las velocidades de línea. En el caso peor, los paquetes entrantes por cada uno de los n puertos de entrada estarán destinados al *mismo* puerto de salida. En dicho caso, durante el tiempo que se tarda en recibir (o enviar) un único paquete, llegarán n paquetes a este puerto de salida. Dado que el puerto de salida sólo puede transmitir un único paquete en una unidad de tiempo (el tiempo de transmisión de paquete), los n paquetes entrantes se pondrán en cola (a esperar) para poder ser transmitidos por el enlace de salida. A continuación, posiblemente puedan llegar n paquetes más en el tiempo que se tarda en transmitir uno solo de los n paquetes que previamente se han introducido en la cola, y así sucesivamente. Finalmente, el número de paquetes en cola podría crecer lo suficiente como para agotar el espacio de memoria en el puerto de salida, en cuyo caso los paquetes serán descartados.

La cola del puerto de salida se muestra en la Figura 4.10. En el instante t ha llegado un paquete a cada uno de los puertos de entrada, y todos ellos están destinados al puerto de salida situado más arriba. Suponiendo velocidades de línea idénticas y un conmutador operando a tres veces la velocidad de línea, una unidad de tiempo después (es decir, en el tiempo que se necesita para recibir o enviar un paquete) los tres paquetes originales habrán sido transferidos al puerto de salida y estarán en cola esperando a ser transmitidos. En la siguiente unidad de tiempo, uno de estos tres paquetes habrá sido transmitido a través del enlace de salida. En nuestro ejemplo, llegan dos *nuevos* paquetes a la entrada del dispositivo de conmutación y uno de estos paquetes también tiene como destino el puerto de salida de más arriba.

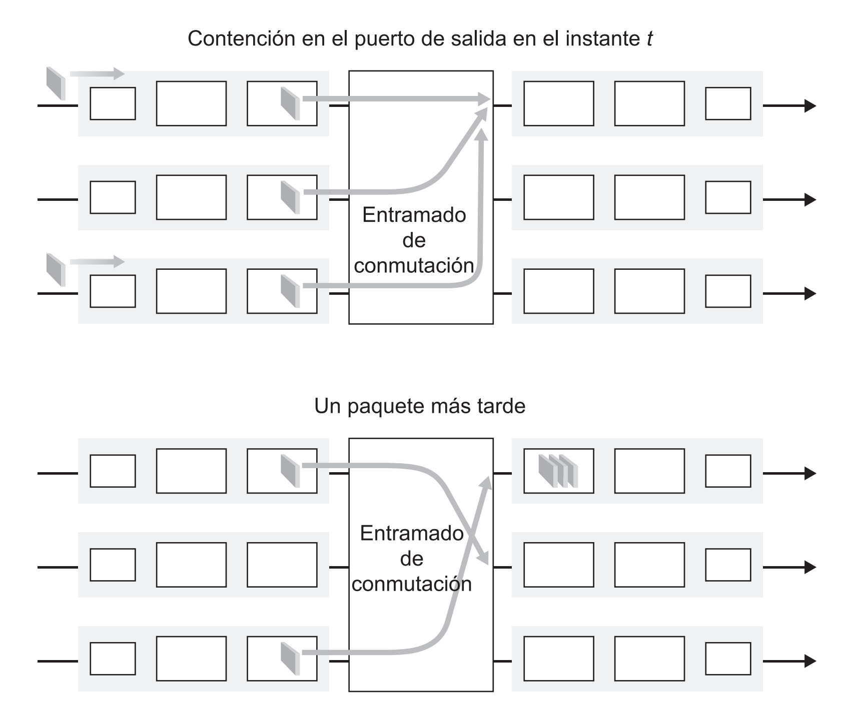


Figura 4.10 • Puesta en cola en el puerto de salida.

Dado que los buffers de los routers son necesarios para absorber las fluctuaciones de la carga de tráfico, es natural plantearse *qué espacio* de buffer será necesario. Durante muchos años, la regla heurística [RFC 3439] que se empleaba para determinar el tamaño del buffer era que la cantidad de espacio de buffer *(B)* debía ser igual al valor medio del tiempo de ida y vuelta (*RTT*, digamos 250 milisegundos) por la capacidad del enlace *(C)*. Este resultado está basado en el análisis de la dinámica de colas de un número relativamente pequeño de flujos TCP [Villamizar 1994]. Por tanto, un enlace a 10 Gbps con un RTT de 250 milisegundos necesitaría un espacio de buffer igual a $B = RTT \cdot C$ = 2,5 Gbits. Sin embargo, recientes esfuerzos teóricos y experimentales [Appenzeller 2004] sugieren que cuando existe un número grande de flujos TCP (*N*) atravesando un enlace, la cantidad de espacio en buffer necesaria es $B = RTT \quad C/\ \overline{N}$. Con una gran cantidad de flujos atravesando los enlaces de router troncales (véase por ejemplo [Fraleigh 2003]), el valor de *N* puede ser grande, con lo que la reducción del tamaño de buffer necesario se hace bastante significativa. [Appenzellar 2004; Wischik 2005; Beheshti 2008] proporcionan una serie de estudios comprensibles acerca del problema del tamaño del buffer desde los puntos de vista teórico, de implementación y operacional.

Una consecuencia de las colas de los puertos de salida es que un **planificador de paquetes** en dicho puerto deberá elegir un paquete de entre aquellos que están en cola esperando a ser transmitidos. Esta selección podría llevarse a cabo basándose en algo simple, como por ejemplo una planificación FCFS (*First-Come-First-Served*, el primero que llega es el primero que se sirve), o en una planificación más sofisticada, como por ejemplo las colas ponderadas equitativas (WFQ, *Weighted Fair Queuing*), que comparten el enlace de salida equitativamente entre las distintas conexiones terminal a terminal que tienen paquetes en cola esperando a ser transmitidos. La planificación de paquetes desempeña un papel fundamental en proporcionar las **garantías de la calidad del servicio**. En el Capítulo 7 se cubre ampliamente la planificación de paquetes. Puede encontrar una exposición sobre las disciplinas de planificación de paquetes en los puertos de salida en [Cisco Queue 2009].

De forma similar, si no hay disponible bastante memoria para almacenar en buffer un paquete entrante, tiene que tomarse la decisión bien de descartar dicho paquete (una política conocida como ***drop-tail*** o eliminación del último) o bien de eliminar uno o más paquetes de los que ya están en cola para hacer sitio al paquete que acaba de llegar. En algunos casos, puede ser una ventaja descartar (o marcar la cabecera de) un paquete *antes* de que el buffer esté lleno, con el fin de proporcionar una indicación de congestión al emisor. En [Labrador 1999, Hollot 2002] se proponen y analizan una serie de políticas para descartar y marcar paquetes, que colectivamente se conocen como algoritmos **AQM (*Active Queue Management*, Gestión activa de colas**). Uno de los algoritmos AQM más ampliamente estudiados e implementados es el algoritmo **RED (*Random Early Detection*, Detección aleatoria temprana**). Con RED se mantiene una media ponderada de la longitud de la cola de salida. Si la longitud media de la cola es menor que un umbral mínimo, min_{th}, cuando llega un paquete éste es admitido en la cola. Inversamente, si la cola está llena o la longitud media de la misma es mayor que un umbral máximo, max_{th}, cuando llega un paquete éste se marca o se descarta. Por último, si llega un paquete y la longitud media de la cola se encuentra dentro del intervalo $[min_{th}, max_{th}]$, el paquete se marca o se descarta con una probabilidad que normalmente es una función de la longitud media de la cola, de min_{th} y de max_{th}. Se han propuesto diversas funciones probabilísticas de marcado/descarte y varias versiones de RED han sido modeladas analíticamente, simuladas y/o implementadas. [Christiansen 2001] y [Floyd 2009] proporcionan panorámicas y referencias a lecturas adicionales.

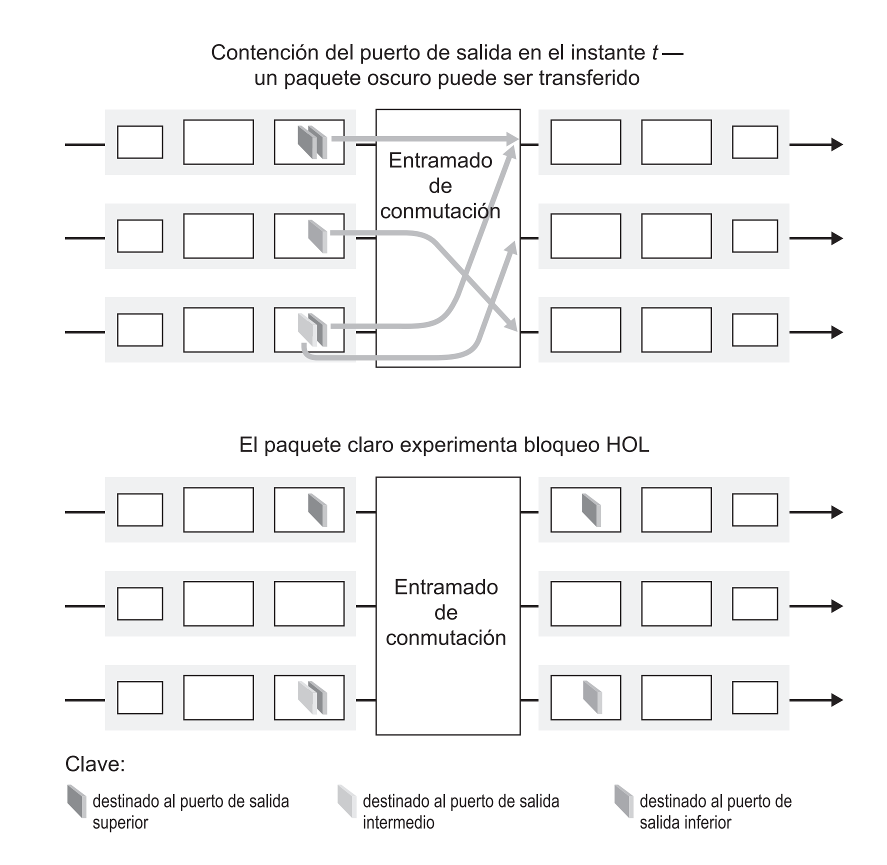


Figura 4.11 • Bloqueo HOL en una cola de entrada de un conmutador.

Si el entramado de conmutación no es lo suficiente rápido (respecto a las velocidades de línea de entrada) como para transferir *todos* los paquetes que le llegan sin producir retardos, entonces también pueden crearse colas de paquetes en los puertos de entrada, ya que los paquetes se irán añadiendo a las colas de los puertos de entrada con el fin de esperar su turno para ser transferidos a través del entramado de conmutación hacia el puerto de salida. Para ilustrar una importante consecuencia de estas colas, considere una estructura de conmutadores en malla y suponga que (1) todas las velocidades de línea son idénticas, (2) que un paquete puede ser transferido desde cualquier puerto de entrada a un puerto de salida dado en el mismo intervalo de tiempo que tarda un paquete en ser recibido a través de un enlace de entrada y (3) que los paquetes pasan de una cola de entrada dada a la cola de salida deseada siguiendo una planificación FCFS. Múltiples paquetes pueden ser transferidos en paralelo, siempre y cuando sus puertos de salida sean diferentes. Sin embargo, si dos paquetes situados en primer lugar de dos colas de entrada están ambos destinados a la misma cola de salida, entonces uno de los paquetes quedará bloqueado y tendrá que esperar en la cola de entrada (el entramado de conmutación sólo puede transferir un paquete a un determinado puerto de salida cada vez).

La Figura 4.11 muestra un ejemplo en el que dos paquetes (los más oscuros) se encuentran al frente de sus respectivas colas de entrada y ambos están destinados al mismo puerto de salida (el de más arriba). Suponga que el entramado de conmutación elige transferir el

paquete de la cabecera de la cola superior izquierda. En este caso, el paquete más oscuro de la cola inferior izquierda tendrá que esperar. Pero no sólo este paquete más oscuro tiene que esperar, sino también el paquete sombreado más claro que está en la cola detrás del paquete de la cola inferior izquierda, incluso aunque *no* exista contención para el puerto de salida intermedio (que es el destino del paquete más claro). Este fenómeno se conoce como **bloqueo HOL (*Head-of-the-line*, Cabeza de la línea**) en un conmutador con cola de entrada (un paquete que se encuentra en una cola de entrada tiene que esperar a ser transferido a través del entramado de conmutación, aunque su puerto de salida esté libre, porque está bloqueado por otro paquete que se encuentra en la cabeza de la línea). [Karol 1987] demuestra que, a causa del bloqueo HOL, la cola de entrada crecerá hasta una longitud ilimitada (informalmente, esto es equivalente a decir que se producirá una pérdida de paquetes significativa) bajo ciertas suposiciones, en cuanto la tasa de llegada de paquetes a través de los enlaces de entrada alcance el 58 por ciento de su capacidad. En [McKeown 1997b] se exponen una serie de soluciones para el bloqueo HOL.

4.4 Protocolo de Internet (IP): reenvío y direccionamiento en Internet

Hasta aquí no hemos hecho referencia a ninguna red de computadoras específica al hablar del reenvío y el direccionamiento de la capa de red. En esta sección vamos a fijar nuestra atención en cómo se llevan a cabo las tareas de reenvío y direccionamiento en Internet. Veremos que tanto el reenvío como el direccionamiento en Internet son componentes importantes del Protocolo de Internet (IP). Actualmente hay dos versiones en uso de IP. En primer lugar, examinaremos el ampliamente utilizado protocolo IP versión 4, que habitualmente se denomina IPv4 [RFC 791]. Al final de la sección, abordaremos la versión 6 de IP [RFC 2460; RFC 4291], protocolo propuesto para sustituir a IPv4.

Pero antes de entrar de lleno en IP, demos un paso atrás y consideremos los componentes que forman la capa de red de Internet. Como se muestra en la Figura 4.12, la capa de red de Internet tiene tres componentes principales. El primero de ellos es el protocolo IP, que es el tema de esta sección. El segundo es el componente de enrutamiento, el cual determina la ruta que sigue un datagrama desde el origen al destino. Hemos mencionado anteriormente que los protocolos de enrutamiento calculan las tablas de reenvío que se utilizan para transmitir los paquetes a través de la red. Estudiaremos los protocolos de enrutamiento de Internet en la Sección 4.6. El último componente de la capa de red es una facilidad que permite informar de la existencia de errores en los datagramas y contesta a las solicitudes de determinada información de la capa de red. En la Sección 4.4.3 abordaremos el protocolo de información de control y de errores de la capa de red de Internet, el protocolo ICMP (*Internet Control Message Protocol*).

4.4.1 Formato de los datagramas

Recuerde que nos referiremos a los paquetes de la capa de red como *datagramas*. Iniciamos nuestro estudio del protocolo IP con una introducción a la sintaxis y la semántica del datagrama de IPv4. Es posible que esté pensando que no puede haber nada más árido que la sintaxis y la semántica de los bits de un paquete. Sin embargo, los datagramas desempeñan un papel central en Internet: todos los estudiantes y profesionales de las redes necesitan com-

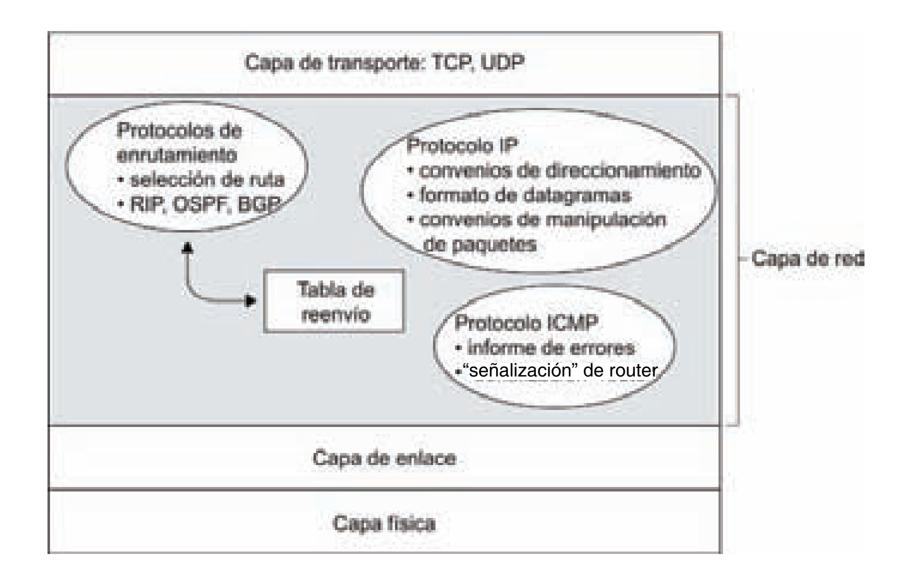


Figura 4.12 • Capa de red de Internet.

prenderlos y dominarlos. El formato de los datagramas de IPv4 se muestra en la Figura 4.13. Los campos clave de los datagramas de IPv4 son los siguientes:

- *Número de versión.* Estos 4 bits especifican la versión del protocolo IP del datagrama. A partir del número de versión el router puede determinar cómo interpretar el resto del datagrama IP. Las distintas versiones de IP utilizan distintos formatos de datagrama. El formato de datagrama de la versión actual de IP, IPv4, es el mostrado en la Figura 4.13.

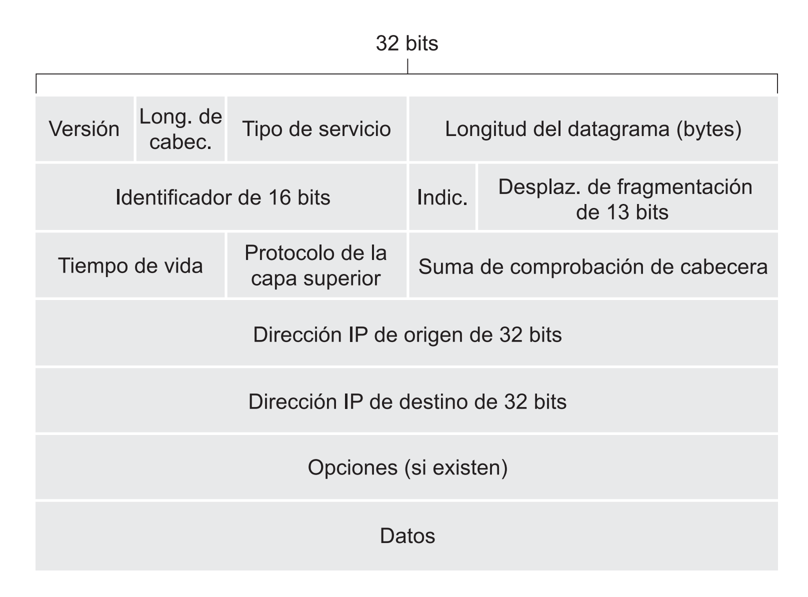


Figura 4.13 • Formato de los datagramas de IPv4.

Al final de la sección veremos el formato de datagrama correspondiente a la nueva versión de IP (IPv6).

- *Longitud de cabecera.* Puesto que un datagrama IPv4 puede contener un número variable de opciones (las que se incluyen en la cabecera del datagrama IPv4), estos 4 bits son necesarios para determinar dónde comienzan realmente los datos del datagrama IP. La mayoría de los datagramas IP no contienen opciones, por lo que el datagrama IP típico tiene una cabecera de 20 bytes.
- *Tipo de servicio.* Los bits del tipo de servicio (TOS, *Type of service*) se incluyeron en la cabecera de IPv4 con el fin de poder diferenciar entre distintos tipos de datagramas IP (por ejemplo, datagramas que requieran en particular un bajo retardo, una alta tasa de transferencia o una entrega fiable). Por ejemplo, puede resultar útil diferenciar datagramas en tiempo real (como los utilizados en aplicaciones de telefonía IP) del tráfico que no es en tiempo real (como por ejemplo el tráfico FTP). El nivel específico de servicio que se proporcione es una política que determinará el administrador del router. Estudiaremos en detalle el tema de los servicios diferenciados en el Capítulo 7.
- *Longitud del datagrama.* Es la longitud total del datagrama IP (la cabecera más lo datos) en bytes. Puesto que este campo tiene una longitud de 16 bits, el tamaño máximo teórico del datagrama IP es de 65.535 bytes. Sin embargo, los datagramas rara vez tienen una longitud mayor de 1.500 bytes.
- *Identificador, indicadores, desplazamiento de fragmentación.* Estos tres campos tienen que ver con lo que se denomina fragmentación IP, un tema que estudiaremos en profundidad enseguida. Es interesante comentar que la nueva versión de IP, IPv6, no permite la fragmentación en los routers.
- *Tiempo de vida.* El campo Tiempo de vida (TTL, *Time-To-Live*) se incluye con el fin de garantizar que los datagramas no estarán eternamente en circulación a través de la red (debido, por ejemplo, a un bucle de enrutamiento de larga duración). Este campo se decrementa en una unidad cada vez que un router procesa un datagrama. Si el campo TTL alcanza el valor 0, el datagrama tiene que ser descartado.
- *Protocolo.* Este campo sólo se emplea cuando un datagrama IP alcanza su destino final. El valor de este campo indica el protocolo específico de la capa de transporte al que se pasarán los datos contenidos en ese datagrama IP. Por ejemplo, un valor de 6 indica que los datos se pasan a TCP, mientras que un valor igual a 17 indica que los datos se pasan a UDP. Puede obtener una lista de todos los valores posibles en [IANA Protocol Numbers 2009]. Observe que el número de protocolo especificado en el datagrama IP desempeña un papel análogo al del campo que almacena el número de puerto de un segmento de la capa de transporte. El número de protocolo es el elemento que enlaza las capas de red y de transporte, mientras que el número de puerto es el componente que enlaza las capas de transporte y de aplicación. En el Capítulo 5 veremos que la trama de la capa de enlace también contiene un campo especial que enlaza la capa de enlace con la capa de red.
- *Suma de comprobación de cabecera.* La suma de comprobación de cabecera ayuda a los routers a detectar errores de bit en un datagrama IP recibido. Esta suma de comprobación se calcula tratando cada pareja de 2 bytes de la cabecera como un número y sumando dichos números utilizando aritmética de complemento a 1. Como se ha visto en la Sección 3.3, el complemento a 1 de esta suma, conocida como suma de comprobación Internet, se almacena en el campo Suma de comprobación. Un router calcula la suma de comprobación de cabecera para cada datagrama IP recibido y detecta una condición

de error si la suma de comprobación incluida en la cabecera del datagrama no coincide con la suma de comprobación calculada. Normalmente, los routers descartan los datagramas en los que se ha detectado que existe un error. Observe que la suma de comprobación tiene que volver a calcularse y almacenarse en cada router, ya que el campo TTL y, posiblemente, también el campo de opciones pueden cambiar. Una interesante exposición acerca de algoritmos rápidos para el cálculo de la suma de comprobación Internet puede verse en [RFC 1071]. Una cuestión que suele plantearse en este punto es ¿por qué TCP/IP lleva a cabo una comprobación de errores tanto en la capa de transporte como en la capa de red? Existen varias razones para esta redundancia. En primer lugar, fíjese en que en la capa IP sólo se calcula la suma de comprobación para la cabecera IP, mientras que la suma de comprobación TCP/UDP se calcula sobre el segmento TCP/UDP completo. En segundo lugar, TCP/UDP e IP no necesariamente tienen que pertenecer a la misma pila de protocolos. En principio, TCP puede ejecutarse sobre un protocolo diferente (por ejemplo, ATM) e IP puede transportar datos que no se pasarán a TCP/UDP.

- *Direcciones IP de origen y de destino.* Cuando un origen crea un datagrama, inserta su dirección IP en el campo de dirección IP de origen e inserta la dirección del destino final en el campo de dirección IP de destino. A menudo el host de origen determina la dirección de destino mediante una búsqueda DNS, como se ha explicado en el Capítulo 2. En la Sección 4.4.2 veremos en detalle el direccionamiento IP.
- *Opciones.* El campo de opciones permite ampliar una cabecera IP. La idea original era que las opciones de cabecera rara vez se emplearan: de ahí la decisión de ahorrar recursos no incluyendo la información de los campos opcionales en la cabecera de todos los datagramas. Sin embargo, la mera existencia de opciones complica las cosas, ya que las cabeceras de datagrama pueden tener una longitud variable, por lo que no puede determinarse a priori dónde comenzará el campo de datos. Además, dado que algunos datagramas pueden requerir el procesamiento de opciones y otros no, la cantidad de tiempo necesario para procesar un datagrama IP en un router puede variar enormemente. Estas consideraciones cobran una particular importancia en el procesamiento IP realizado en los hosts y routers de altas prestaciones. Por estas razones y otras, las opciones IP fueron eliminadas en la cabecera de IPv6, como veremos en la Sección 4.4.4.
- *Datos (carga útil).* Finalmente, llegamos al último campo y más importante: ¡la *razón de ser* del datagrama! En la mayoría de las circunstancias, el campo de datos del datagrama IP contiene el segmento de la capa de transporte (TCP o UDP) que va a entregarse al destino. Sin embargo, el campo de datos puede transportar otros tipos de datos, como por ejemplo mensajes ICMP (que veremos en la Sección 4.4.3).

Observe que un datagrama IP tiene un total de 20 bytes de cabecera (suponiendo que no contienen opciones). Si el datagrama transporta un segmento TCP, entonces cada datagrama (no fragmentado) transporta un total de 40 bytes de cabecera (20 bytes de la cabecera IP más 20 bytes de la cabecera TCP) junto con el mensaje de la capa de aplicación.

Fragmentación del datagrama IP

En el Capítulo 5 veremos que no todos los protocolos de la capa de enlace pueden transportar paquetes de la capa de red del mismo tamaño. Algunos protocolos pueden transportar datagramas grandes, mientras que otros sólo transportan datagramas pequeños. Por ejemplo, las tramas Ethernet pueden transportar hasta 1.500 bytes de datos, mientras que las tramas

para otros enlaces de área extensa no pueden transportar más de 576 bytes. La cantidad máxima de datos que una trama de la capa de enlace puede transportar se conoce como unidad máxima de transmisión (MTU, *Maximum Transmission Unit*). Puesto que cada datagrama IP se encapsula dentro de una trama de la capa de enlace para ir de un router al siguiente, la MTU del protocolo de la capa de enlace impone un límite estricto a la longitud de un datagrama IP. Esta limitación del tamaño de un datagrama IP no supone un problema importante. Lo que realmente es un problema es que cada uno de los enlaces existentes a lo largo de la ruta entre el emisor y el destino pueden utilizar diferentes protocolos de la capa de enlace y cada uno de estos protocolos puede utilizar una MTU diferente.

Con el fin de comprender mejor la cuestión de reenvío, imagine que *usted* es un router que interconecta varios enlaces, que ejecutan distintos protocolos de la capa de enlace con MTU diferentes. Suponga que recibe un datagrama IP procedente de un enlace. Comprueba su tabla de reenvío para determinar el enlace de salida y ese enlace de salida tiene una MTU que es menor que la longitud del datagrama IP. Lo que nos lleva a plantearnos la pregunta de cómo meter ese datagrama IP sobredimensionado en el campo de carga útil de la trama de la capa de enlace. La solución consiste en fragmentar los datos del datagrama IP en dos o más datagramas IP más pequeños, encapsular cada uno de los datagramas IP más pequeños en una trama de la capa de enlace distinta y enviar dichas tramas a través del enlace de salida. Cada uno de estos datagramas más pequeños se conocen como **fragmentos**.

Los fragmentos tienen que ser reensamblados antes de llegar a la capa de transporte del destino. De hecho, tanto TCP como UDP están esperando recibir segmentos completos no fragmentos de la capa de red. Los diseñadores de IPv4 pensaron que reensamblar los datagramas en los routers añadiría una complejidad significativa al protocolo y reduciría su rendimiento. (Si usted fuera un router, ¿querría reensamblar fragmentos además de todo lo que ya tiene que hacer?) Siguiendo el principio de mantener el núcleo de la red simple, los diseñadores de IPv4 decidieron dar el trabajo de reensamblar los datagramas a los sistemas terminales en lugar de a los routers de red.

Cuando un host de destino recibe una serie de datagramas procedentes del mismo origen, tiene que determinar si alguno de esos datagramas son fragmentos de algún otro datagrama original más grande. Si algunos datagramas son fragmentos, tiene que determinar además cuándo ha recibido el último fragmento y cómo debe ensamblar los fragmentos que ha recibido para formar el datagrama original. Para que el host de destino pueda llevar a cabo estas tareas de reensamblado, los diseñadores de IP (versión 4) incluyeron los campos *identificación, indicador* y *desplazamiento de fragmentación* en la cabecera del datagrama IP. Cuando se crea un datagrama, el host emisor marca el datagrama con un número de identificación, así como con las direcciones de origen y de destino. Normalmente, el host emisor incrementa el número de identificación para cada datagrama que envía. Cuando un router necesita fragmentar un datagrama, cada datagrama resultante (es decir, cada fragmento) se marca con la dirección de origen, la dirección de destino y el número de identificación del datagrama original. Cuando el destino recibe una serie de datagramas procedentes del mismo host emisor, puede examinar los números de identificación de los datagramas para determinar cuáles de ellos son fragmentos de un mismo datagrama más largo. Puesto que IP es un servicio no fiable, es posible que uno o más de los fragmentos nunca lleguen a su destino. Por esta razón, con el fin de que el host de destino esté absolutamente seguro de que ha recibido el último fragmento del datagrama original, el último fragmento tiene un bit indicador puesto a 0, mientras que los demás fragmentos tienen el bit indicador puesto a 1. Además, para que el host de destino deter-

mine si falta un fragmento (y también para que pueda reensamblar los fragmentos en el orden apropiado), se utiliza el campo desplazamiento para especificar en qué posición dentro del datagrama IP original encaja el fragmento.

La Figura 4.14 ilustra un ejemplo. Un datagrama de 4.000 bytes (20 bytes de cabecera IP más 3.980 bytes de carga útil IP) llega a un router y tienen que ser reenviado a un enlace con una MTU de 1.500 bytes. Esto implica que los 3.980 bytes de datos del datagrama original tienen que ser alojados en tres fragmentos distintos (cada uno de los cuales también es un datagrama IP). Suponga que el datagrama original está marcado con el número de identificación 777. Las características de los tres fragmentos se muestran en la Tabla 4.2. Los valores de esta tabla reflejan el requisito de que la cantidad de datos útiles originales de todos los fragmentos, excepto del último, tiene que ser un múltiplo de 8 bytes, y que el valor del desplazamiento debe especificarse en unidades de fragmentos de 8 bytes.

En el destino, la carga útil del datagrama se pasa a la capa de transporte sólo después de que la capa IP haya reconstruido completamente el datagrama IP original. Si uno o más de los fragmentos no llegan al destino, el datagrama incompleto se descarta y no se pasa a la capa de transporte. Pero, como hemos visto en el capítulo anterior, si se está utilizando TCP en la capa de transporte, entonces TCP se recuperará de esta pérdida haciendo que el origen retransmita los datos del datagrama original.

Acabamos de ver que la fragmentación IP desempeña un papel importante en el ensamblado de las muchas y dispares tecnologías de la capa de enlace. Pero la fragmentación también tiene sus costes. En primer lugar, añade complejidad a los routers y sistemas terminales, que tienen que ser diseñados para acomodar la fragmentación y el reensamblado de los data

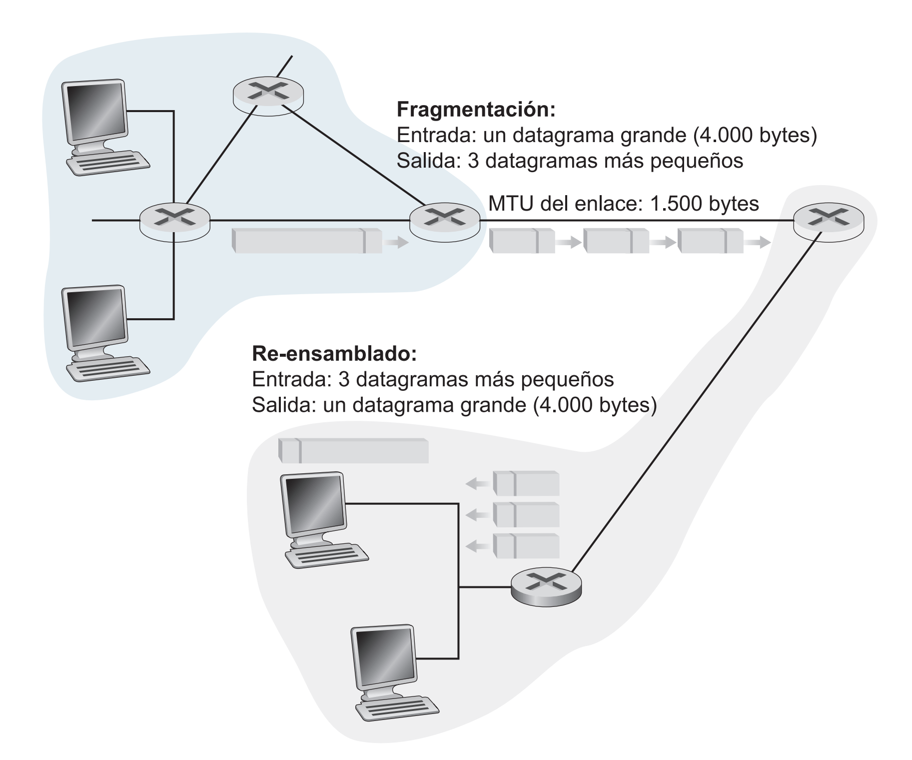


Figura 4.14 • Fragmentación y reensamblado IP.

Fragmento	Bytes	ID	Desplazamiento	Indicador
Primer fragmento	1.480 bytes en el campo de datos del datagrama IP	Identificación 777	Desplaz. 0 (significa que los datos deben insertarse comenzando en el byte 0)	Indicador 1 (hay más fragmentos)
Segundo fragmento	1.480 bytes de datos	Identificación 777	Desplaz. 185 (significa que los datos deben insertarse comenzando en el byte 1.480. Observe que 185 · 8 1.480)	Indicador 1 (hay más fragmentos)
Tercer fragmento	1.020 bytes (3.980–1.480 –1.480)de datos	Identificación 777	Desplaz. 370 (significa que los datos deben insertarse comenzando en el byte 2.960. Observe que 370 · 8 2.960)	Indicador 0 (es el último fragmento)

Tabla 4.2 • Fragmentos IP.

gramas. En segundo lugar, la fragmentación se puede utilizar para crear ataques DoS letales, en los que el atacante envía una serie de fragmentos extraños e inesperados. Un ejemplo clásico es el ataque Jolt2, en el que el atacante envía un flujo de fragmentos pequeños al host objetivo, ninguno de los cuales tiene un desplazamiento igual a cero. El host objetivo puede colapsar al intentar reconstruir datagramas a partir de los paquetes degenerados. Otro tipo de ataque envía fragmentos IP solapados, es decir, fragmentos cuyos valores de desplazamiento se han establecido de modo que los fragmentos no se alinean apropiadamente. Los sistemas operativos vulnerables, que no saben qué hacer con esos fragmentos solapados, pueden dejar de funcionar [Skoudis 2006]. Como veremos al final de esta sección, la nueva versión del protocolo IP, IPv6, no admite la fragmentación, simplificando el procesamiento de paquetes IP y haciendo que IP sea menos vulnerable a los ataques.

En el sitio web del libro proporcionamos un applet de Java que genera fragmentos. No tiene más que proporcionar el tamaño del datagrama de entrada, la MTU y la identificación del datagrama de entrada y el applet genera automáticamente los fragmentos. Visite el sitio http:// www.awl.com/kurose-ross.

4.4.2 Direccionamiento IPv4

Ahora vamos a ocuparnos del direccionamiento IPv4. Aunque puede que piense que el direccionamiento es un tema sencillo, tenemos la esperanza de que al terminar este capítulo se haya convencido de que el direccionamiento en Internet no sólo es un tema interesante y sutil, sino que también tiene una gran importancia para Internet. Tanto [3Com Addressing 2009] como el primer capítulo de [Stewart 1999] proporcionan un excelente tratamiento del direccionamiento IPv4.

Sin embargo, antes de abordar el direccionamiento IP, necesitamos dedicar unas pocas palabras a cómo los hosts y los routers están conectados en la red. Normalmente, un host dispone de un único enlace hacia la red; cuando IP en el host desea enviar un datagrama, lo

hace a través de este enlace. El límite entre el host y el enlace físico se denomina **interfaz**. Consideremos a continuación un router y sus interfaces. Puesto que la tarea de un router consiste en recibir un datagrama por un enlace y reenviarlo a algún otro enlace, un router necesariamente está conectado a dos o más enlaces. El límite entre el router y cualquiera de sus enlaces también se conoce como interfaz. Por tanto, un router tiene varias interfaces, una para cada uno de los enlaces. Puesto que todos los hosts y todos los routers son capaces de enviar y recibir datagramas IP, IP requiere que cada interfaz de host y de router tenga su propia dirección IP. Por tanto, técnicamente, una dirección IP está asociada con una interfaz, en lugar de con el host o con el router que contiene dicha interfaz.

Las direcciones IP tienen una longitud de 32 bits (lo que equivale a 4 bytes), por lo que existen un total de 2^{32} direcciones IP posibles. Aproximando 2^{10} a 10^3, es fácil ver que hay unos 4.000 millones direcciones IP posibles. Estas direcciones normalmente se expresan utilizando la **notación decimal con punto**, en la que cada byte de la dirección se escribe en formato decimal y separada mediante un punto del resto de los bytes de la dirección. Por ejemplo, considere la dirección IP 193.32.216.9. El 193 es el número decimal equivalente a los 8 primeros bits de la dirección; el 32 es el equivalente decimal de los segundos 8 bits de la dirección, y así sucesivamente. Por tanto, la dirección 193.32.216.9 en notación binaria se expresa como sigue:

11000001 00100000 11011000 00001001

Cada una de las interfaces de un host o de un router de Internet tiene que tener asociada una dirección IP que es globalmente única (excepto en el caso de las interfaces utilizadas para NAT, que veremos al final de esta sección). No obstante, estas direcciones no se pueden elegir a tontas y a locas. Una parte de la dirección IP de una interfaz estará determinada por la subred a la que está conectada.

La Figura 4.15 proporciona un ejemplo de las interfaces y del direccionamiento IP. En esta figura, se utiliza un router (con tres interfaces) para interconectar siete hosts. Echemos

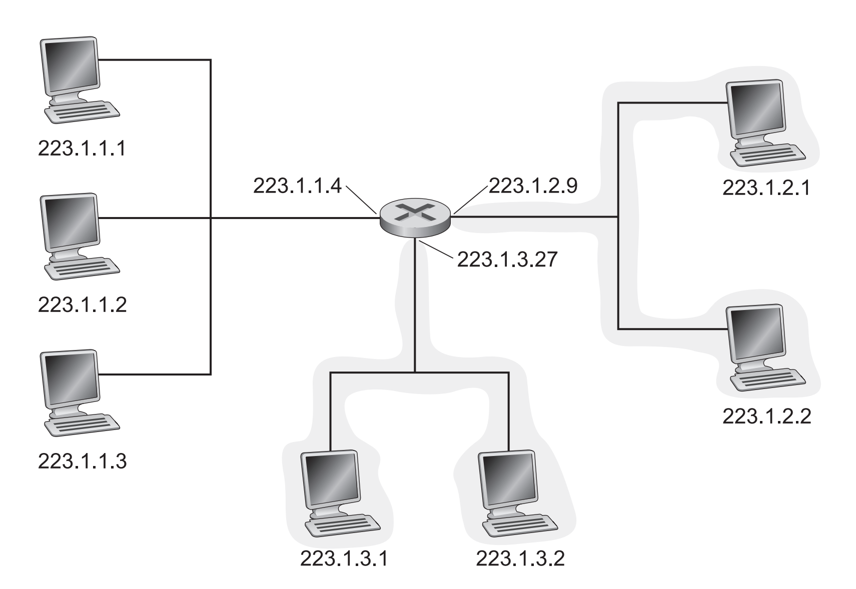


Figura 4.15 • Subredes y direcciones de las interfaces.

un vistazo a las direcciones IP asignadas a las interfaces de los hosts y del router; hay varios puntos que merece la pena destacar. Los tres hosts de la parte superior izquierda de la Figura 4.15 y la interfaz del router a la que están conectados, todos ellos tienen una dirección IP con el formato 223.1.1.xxx. Es decir, los 24 bits más a la izquierda de la dirección IP de todos ellos son iguales. Además, las cuatro interfaces están interconectadas mediante una red *que no contiene routers*. (Esta red podría ser, por ejemplo una LAN Ethernet, en cuyo caso las interfaces estarían interconectadas mediante un hub Ethernet o un switch Ethernet; véase el Capítulo 5.) En términos de IP, esta red que interconecta tres interfaces de host y una interfaz de router forma una **subred** [RFC 950]. (Una subred también se conoce como *red IP* o simplemente *red* en la literatura dedicada a Internet.) El direccionamiento IP asigna una dirección a esta subred: 223.1.1.0/24, donde la notación /24, que en ocasiones se denomina **máscara de subred**, indica que los 24 bits más a la izquierda de la magnitud de 32 bits define la dirección de subred. Por tanto, la subred 223.1.1.0/24 consta de las tres interfaces de host (223.1.1.1, 223.1.1.2 y 223.1.1.3) y de la interfaz del router (223.1.1.4). Cualquier host adicional conectado a la subred 223.1.1.0/24 *requeriría* una dirección de la forma 223.1.1.xxx. En la Figura 4.15 se muestran otras dos subredes adicionales: la red 223.1.2.0/24 y la subred 223.1.3.0/24. La Figura 4.16 ilustra las tres subredes IP presentes en la Figura 4.15.

La definición IP de una subred no está restringida a los segmentos Ethernet que conectan varios hosts a una interfaz de un router. Profundicemos un poco en esta cuestión. Considere la Figura 4.17, que muestra tres routers interconectados entre sí mediante enlaces punto a punto. Cada router tiene tres interfaces, una para cada enlace punto a punto y una para el enlace de difusión que conecta directamente el router a una pareja de hosts. ¿Qué subredes hay presentes aquí? Hay tres: 223.1.1.0/24, 223.1.2.0/24 y 223.1.3.0/24, y son similares a las subredes de la Figura 4.15. Pero fíjese en que, en este ejemplo, también existen tres subredes adicionales: una subred, 223.1.9.0/24, para las interfaces que conectan los routers R1 y R2; otra subred, 223.1.8.0/24, para las interfaces que conectan los routers R2 y R3; y

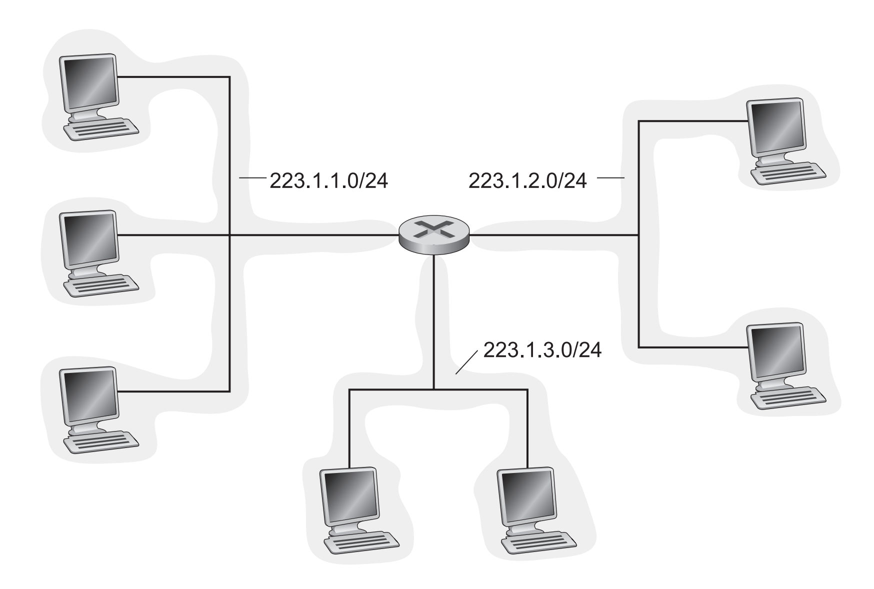


Figura 4.16 • Direcciones de subred.

una tercera subred, 223.1.7.0/24, para las interfaces que conectan los routers R3 y R1. En un sistema interconectado general de routers y hosts, podemos utilizar la siguiente receta para definir las subredes existentes en el sistema:

> *Para determinar las subredes, desconecte cada interfaz de su host o router, creando islas de redes aisladas, con interfaces que acaban en los puntos terminales de las redes aisladas. Cada una de estas redes aisladas se dice que es una* ***subred****.*

Si aplicamos este procedimiento al sistema interconectado de la Figura 4.17, obtenemos seis islas o subredes.

A partir de la exposición anterior, está claro que una organización (como por ejemplo una empresa o una institución académica) con múltiples segmentos Ethernet y enlaces punto a punto tendrá varias subredes, teniendo todos los dispositivos de una subred dada la misma dirección de subred. En principio, las distintas subredes podrían tener direcciones de subred bastante diferentes. Sin embargo, en la práctica, sus direcciones de subred a menudo tienen mucho en común. Para entender por qué, veamos cómo se gestiona el direccionamiento en la Internet global.

La estrategia de asignación de direcciones en Internet se conoce como **Enrutamiento entre dominios sin clase (CIDR,** ***Classless Interdomain Routing***) [RFC 4632]. CIDR generaliza la noción de direccionamiento de subred. Al igual que sucede con las subredes IP, la dirección IP de 32 bits se divide en dos partes y de nuevo se expresa en notación decimal con punto como *a.b.c.d/x*, donde *x* indica el número de bits de la primera parte de la dirección.

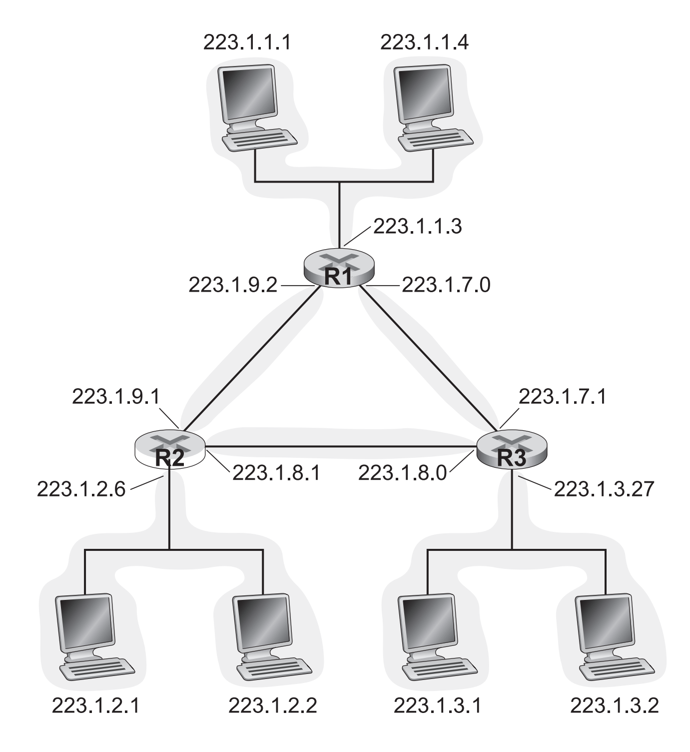


Figura 4.17 • Tres routers que interconectan seis subredes.

Los *x* bits más significativos de una dirección en el formato *a.b.c.d/x* constituyen la parte de red de la dirección IP y a menudo se hace referencia a ellos como el **prefijo** (o *prefijo de red*) de la dirección. Normalmente, una organización tiene asignado un bloque de direciones contiguas; es decir, un rango de direcciones con un prefijo común (véanse los principios en el recuadro Práctica). En este caso, las direcciones IP de los dispositivos que se encuentran dentro de la organización compartirán el mismo prefijo. Cuando estudiemos el protocolo de enrutamiento BGP de Internet en la Sección 4.6, veremos que los routers externos a la red de la organización sólo tienen en cuenta estos *x* primeros bits del prefijo. Es decir, cuando un router externo a la organización reenvía un datagrama cuya dirección de destino está dentro de la organización, únicamente necesita tener en cuenta los primeros *x* bits de la dirección. Esto reduce considerablemente el tamaño de la tabla de reenvío de los routers, ya que una *única* entrada con el formato *a.b.c.d/x* bastará para reenviar paquetes a *cualquier* destino dentro de la organización.

Los 32-*x* bits restantes de una dirección pueden emplearse para diferenciar los dispositivos *internos* de la organización, teniendo todos ellos el mismo prefijo de red. Estos son los



PRÁCTICA

Este ejemplo de un ISP que conecta a ocho organizaciones a Internet ilustra cómo la asignación cuidadosa de direcciones CIDR facilita el enrutamiento. Suponga, como se muestra en la Figura 4.18, que el ISP (al que llamaremos ISP A) anuncia al mundo exterior que enviará cualquier datagrama cuyos primeros 20 bits de dirección se correspondan con 200.23.16.0/20. El resto del mundo no necesita saber que dentro del bloque de direcciones 200.23.16.0/20 existen en realidad ocho organizaciones, cada una con sus propias subredes. Esta capacidad de emplear un mismo prefijo para anunciar múltiples redes suele denominarse **agregación de direcciones** (o **agregación de rutas**, o también **resumen de rutas**).

La técnica de agregación de direcciones funciona extraordinariamente bien cuando las direcciones se asignan en bloques a los ISP y éstos las asignan en bloque a las organizaciones cliente. Pero, ¿qué ocurre si las direcciones no están asignadas de esa forma jerárquica? ¿Qué ocurriría, por ejemplo, si el ISP A adquiere el ISP B y luego hace que la Organización 1 se conecte a Internet a través del ISP B subsidario? Como se muestra en la Figura 4.18, el ISP B subsidario posee el bloque de direcciones 199.31.0.0/16, pero las direcciones IP de la Organización 1 lamentablemente no pertenecen a este bloque de direcciones. ¿Qué habría que hacer en este caso? Realmente, la Organización 1 podría renumerar todos sus routers y hosts para disponer de direcciones contenidas en el bloque de direcciones del ISP B. Pero ésta es una solución costosa y la Organización 1 podría ser reasignada en el futuro a otro ISP subsidiario. La solución que normalmente adoptará la Organización 1 será mantener sus direcciones en 200.23.18.0/23. En este caso, como se muestra en la Figura 4.19, el ISP A continúa anunciando el bloque de direcciones 200.23.16.0/20 y el ISP B continúa anunciando el bloque 199.31.0.0/16. Sin embargo, el ISP B ahora *también* anuncia el bloque de direcciones de la Organización 1, 200.23.18.0/23. Cuando otros routers de Internet vean los bloques de direcciones 200.23.16.0/20 (del ISP A) y 200.23.18.0/23 (del ISP B) y deseen enrutar hacia una dirección contenida en el bloque 200.23.18.0/23, utilizarán la regla de coincidencia con el prefijo más largo (véase la Sección 4.2.2) y lo enviarán hacia el ISP B, ya que anuncia el prefijo de dirección más largo (más específico) que se corresponde con la dirección de destino.

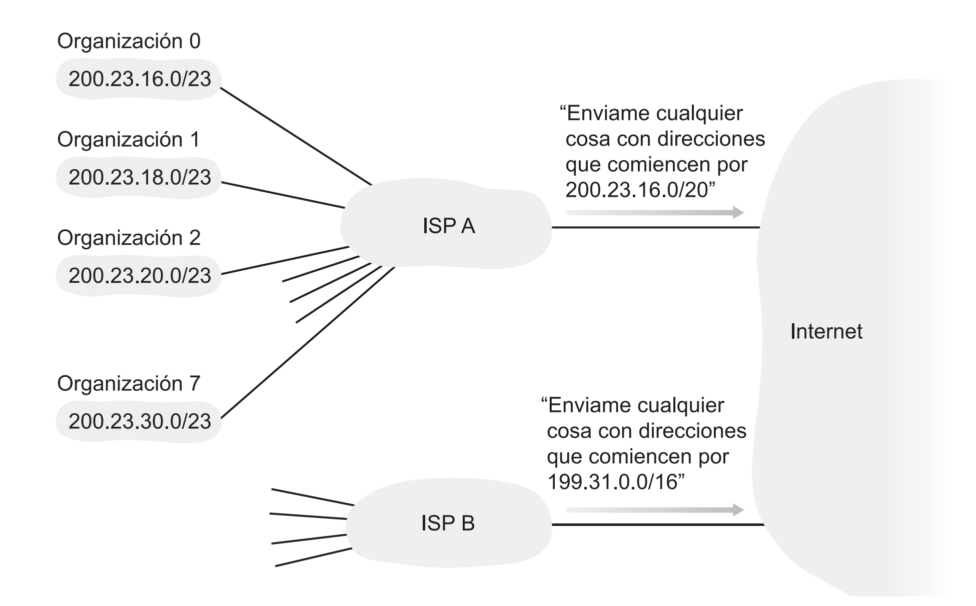


Figura 4.18 • Agregación de rutas y direccionamiento jerárquicos.

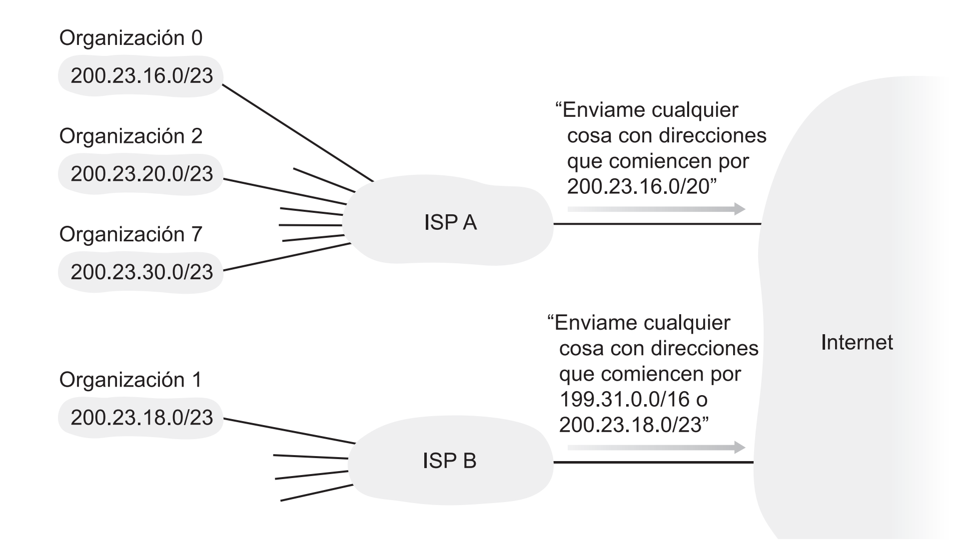


Figura 4.19 • El ISP B tiene una ruta más específica hacia la Organización 1.

bits que habrá que considerar para reenviar paquetes en los routers *internos* de la organización. Estos bits de menor peso pueden tener (o no) una estructura en subred adicional, como la que hemos visto anteriormente. Por ejemplo, suponga que los 21 primeros bits de la dirección CIDR a.b.c.d/21 especifican el prefijo de red de la organización y son comunes a las direcciones IP de todos los dispositivos de dicha organización. Los restantes 11 bits identifican entonces a los hosts específicos de la organización. La estructura interna de la organización puede ser tal que estos 11 bits de más a la derecha se empleen para dividir en

subredes la organización, como hemos viso anteriormente. Por ejemplo, *a.b.c.d/24* podría hacer referencia a una subred específica de la organización.

Antes de que se adoptara el enrutamiento CIDR, la parte de red de una dirección IP estaba restringida a longitudes de 8, 16 o 24 bits, un esquema de direccionamiento conocido como **direccionamiento con clases**, ya que las subredes con direcciones de 8, 16 y 24 bits se conocían, respectivamente, como redes de clase A, B y C. El requisito de que la parte de subred de una dirección IP tuviera exactamente una longitud de 1, 2 o 3 bytes se volvió problemático a la hora de dar soporte al rápido crecimiento de número de organizaciones con subredes de tamaño medio y pequeño. Una subred de clase C (/24) sólo puede acomodar hasta $2^8 - 2 = 254$ hosts (dos de las $2^8 = 256$ direcciones están reservadas para usos especiales), que son muy pocos para muchas organizaciones. Sin embargo, una subred de clase B (/16), que puede dar soporte a 65.634 hosts, era demasiado grande. Con el direccionamiento con clases, una organización con, por ejemplo, 2.000 hosts, era asignada normalmente a una dirección de subred de clase B (/16). Esto llevó a un rápido agotamiento del espacio de direcciones de clase B y a una pobre utilización del espacio de direcciones asignado. Por ejemplo, la organización que empleaba una dirección de clase B para sus 2.000 hosts tenía asignado espacio suficiente para hasta 65.534 interfaces, dejando bloqueadas más de 63.000 direcciones que no podían ser utilizadas por otras organizaciones.

Seríamos negligentes si no mencionáramos que existe otro tipo de dirección IP, la dirección IP de difusión 255.255.255.255. Cuando un host envía un datagrama cuya dirección de destino es 255.255.255.255, el mensaje se entrega a todos los hosts existentes en la misma subred. Opcionalmente, los routers también reenvían el mensaje a las subredes vecinas (aunque habitualmente no lo hacen).

Ahora que hemos estudiado en detalle el direccionamiento IP, necesitamos saber cómo los hosts y las subredes obtienen sus direcciones. Comenzaremos viendo cómo una organización obtiene un bloque de direcciones para sus dispositivos, y luego veremos cómo se asigna una dirección del bloque de direcciones de la organización a un dispositivo (por ejemplo, a un host).

Cómo obtener un bloque de direcciones

Para obtener un bloque de direcciones IP que pueda ser utilizado dentro de la subred de una organización, un administrador de red tiene que contactar en primer lugar con su ISP, el cual le proporcionará direcciones extraidas de un bloque de direcciones mayor que ya habrá sido asignado al ISP. Por ejemplo, al ISP pueden haberle asignado el bloque de direcciones 200.23.16.0/20. A su vez, el ISP podría dividir su bloque de direcciones en ocho bloques de direcciones contiguos del mismo tamaño y asignar cada uno de estos bloques de direcciones a hasta ocho organizaciones a las que puede prestar servicio, como se muestra a continuación (hemos subrayado la parte de subred de estas direcciones).

Bloque del ISP	200.23.16.0/20	11001000 00010111 00010000 00000000
Organización 0	200.23.16.0/23	11001000 00010111 00010000 00000000
Organización 1	200.23.18.0/23	11001000 00010111 00010010 00000000
Organización 2	200.23.20.0/23	11001000 00010111 00010100 00000000
. . .	. . .	. . .
Organización 7	200.23.30.0/23	11001000 00010111 00011110 00000000

Obtener un conjunto de direcciones de un ISP es una forma de conseguir un bloque de direcciones, pero no es la única forma. Evidentemente, también debe existir una forma para el propio ISP de obtener un bloque de direcciones. ¿Existe una entidad autoritativa global cuya responsabilidad última sea gestionar el espacio de direcciones IP y asignar bloques de direcciones a los ISP y otras organizaciones? ¡Por supuesto que existe! Las direcciones IP son gestionadas por la entidad ICANN (*Internet Corporation for Assigned Names and Numbers*, Corporación de Internet para los números y nombres asignados) [ICANN 2009], basándose en las directrices establecidas en [RFC 2050]. El papel de la organización sin ánimo de lucro ICANN [NTIA 1998] no es sólo el de asignar direcciones IP, sino también gestionar los servidores raíz DNS. También tiene el polémico trabajo de asignar nombres de dominio y de resolver las disputas por dichos nombres. La organización ICANN asigna direcciones a los registros regionales de Internet (por ejemplo, ARIN, RIPE, APNIC y LACNIC, que forman la Organización de Soporte de Direcciones de ICANN [ASO-ICANN 2009]), y gestionan la asignación/administración de direcciones dentro de sus regiones.

Cómo obtener una dirección de host: Protocolo de configuración dinámica de host

Una vez que una organización ha obtenido un bloque de direcciones, puede asignar direcciones IP individuales a las interfaces de sus hosts y routers. Normalmente, un administrador de sistemas configura manualmente las direcciones IP de un router (a menudo de forma remota mediante una herramienta de gestión de red). Las direcciones de host también se pueden configurar manualmente, pero frecuentemente ahora esta tarea se lleva cabo utilizando el **Protocolo de configuración dinámica de host (DHCP, *Dynamic Host Configuration Protocol*)** [RFC 2131]. DHCP permite a un host obtener (permite que se le asigne) automáticamente una dirección IP. Un administrador de red puede configurar DHCP de modo que un host dado reciba la misma dirección IP cada vez que se conecte a la red, o un host puede ser asignado a una **dirección IP temporal** que será diferente cada vez que el host se conecte a la red. Además de la asignación de direcciones IP a los hosts, DHCP también permite que un host obtenga información adicional, como por ejemplo su máscara de subred, la dirección del router del primer salto [a menudo denominado router de *pasarela* (*gateway*) predeterminado] y la dirección de su servidor DNS local.

Gracias a la capacidad de DHCP de automatizar el proceso de conexión de un host a una red, a menudo se dice que es un **protocolo plug-and-play**. Esta capacidad le hace *muy* atractivo para el administrador de la red que en otro caso tendría que realizar estas tareas ¡manualmente! DHCP también disfruta de un amplio uso en las redes de acceso a Internet residenciales y en las redes LAN inalámbricas, en las que los hosts se unen a la red y salen de ella frecuentemente. Considere, por ejemplo, un estudiante que traslada una computadora portátil desde su casa a la biblioteca y luego a clase. Probablemente, en cada localización el estudiante se conectará a una subred y, por tanto, necesitará una nueva dirección IP en cada lugar. DHCP está idealmente adaptado para estas situaciones, ya que existen muchos usuarios que van y vienen, y que necesitan direcciones sólo durante un periodo de tiempo limitado. Del mismo modo, DHCP resulta útil en las redes de acceso de los ISP que trabajan en el mercado residencial. Considere por ejemplo un ISP residencial que tiene 2.000 clientes, pero no más de 400 clientes están en línea al mismo tiempo. En este caso, en lugar de necesitar un bloque de 2.048 direcciones, un servidor DHCP que asigne direcciones de forma dinámica sólo necesitará un bloque de 512 direcciones (por ejemplo, un bloque con el for-

mato de dirección *a.b.c.d/23*). A medida que los hosts se unen a la red y salen de ella, el servidor DHCP necesita actualizar su lista de direcciones IP disponibles. Cada vez que un host se une a la red, el servidor DHCP asigna una dirección arbitraria de su conjunto actual de direcciones disponibles; cada vez que un host abandona la red, su dirección es devuelta al conjunto.

DHCP es un protocolo cliente-servidor. Normalmente, un cliente es un host recién llegado que desea obtener información de configuración de la red, incluyendo una dirección IP para sí mismo. En el caso más simple, cada subred (en el sentido de direccionamiento mostrado en la Figura 4.17) tendrá un servidor DHCP. Si en la subred no hay ningún servidor, es necesario un agente de retransmisión DHCP (normalmente un router) que conozca la dirección de un servidor DHCP para dicha red. La Figura 4.20 muestra un servidor DHCP conectado a la subred 223.1.2/24, con el router actuando como agente de retransmisión para los clientes recién llegados que se conectan a las subredes 223.1.1/24 y 223.1.3/24. En la siguiente exposición, supondremos que hay disponible un servidor DHCP en la subred.

Para un host recién llegado a una red, el protocolo DHCP es un proceso de cuatro pasos, como se muestra en la Figura 4.21 para la configuración de red mostrada en la Figura 4.20. En esta figura, `sudirI` ("su dirección Internet") indica la dirección que se asigna al cliente que acaba de llegar. Los cuatro pasos son los siguientes:

- *Descubrimiento del servidor DHCP.* La primera tarea de un host recién llegado es encontrar un servidor DHCP con el que interactuar. Esto se hace mediante un **mensaje de descubrimiento DHCP**, que envía un cliente dentro de un paquete UDP al puerto

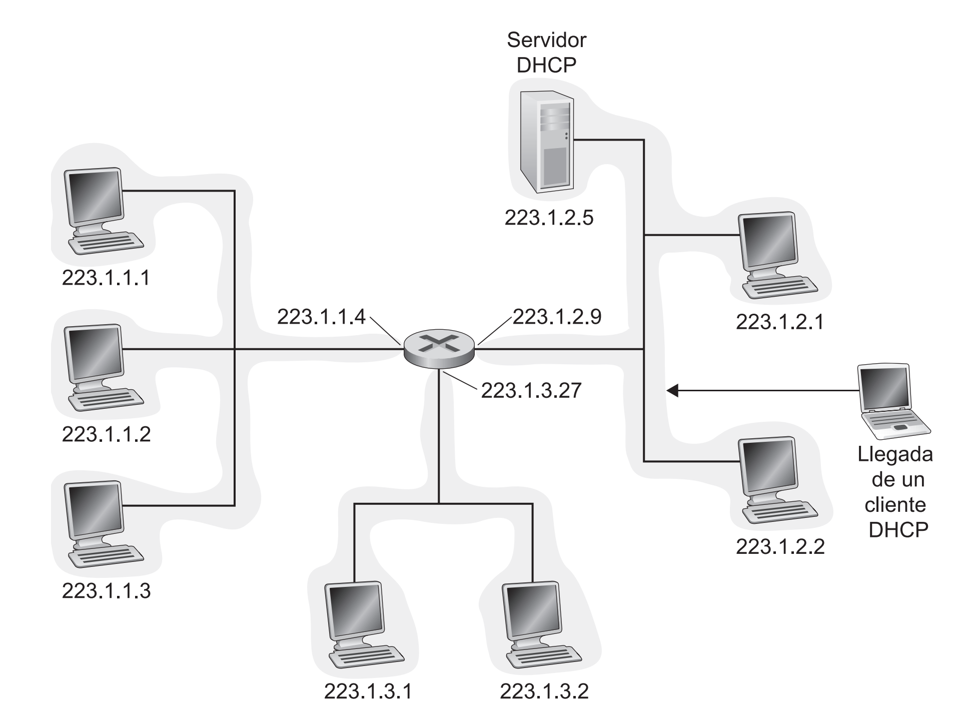


Figura 4.20 • Escenario cliente-servidor DHCP.

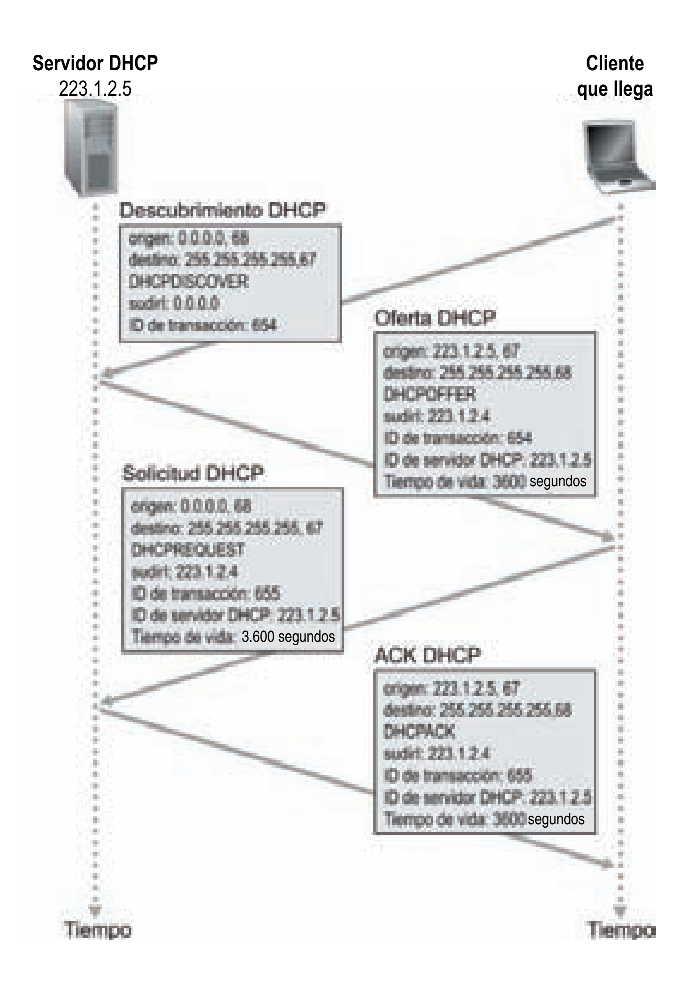


Figura 4.21 • Interacción cliente-servidor DHCP.

67. El paquete UDP se encapsula en un datagrama IP. Pero, ¿a quién debería enviarse este datagrama? El host ni siquiera conoce la dirección IP de la red a la que se está conectando, y mucho menos la dirección de un servidor DHCP de esa red. En esta situación, el cliente DHCP crea un datagrama IP que contiene su mensaje de descubrimiento DHCP junto con la dirección IP de difusión 255.255.255.255 y una dirección IP de origen de "este host" igual a 0.0.0.0. El cliente DHCP pasa el datagrama IP a la capa de enlace, la cual difunde esta trama a todos los nodos conectados a la subred (en la Sección 5.4 estudiaremos en detalle el proceso de difusión de la capa de enlace).

- *Oferta(s) del servidor DHCP.* Un servidor DHCP que recibe un mensaje de descubrimiento DHCP responde al cliente con un **mensaje de oferta DHCP**, que se difunde a todos los nodos de la subred utilizando de nuevo la dirección IP de difusión 255.255.255.255 (es posible que se pregunte por qué la respuesta de este servidor también debe difundirse). Puesto que en la subred pueden existir varios servidores DHCP, el cliente puede encontrarse en la situación envidiable de poder elegir entre varias ofertas. Cada mensaje de oferta de servidor contiene el ID de transacción del mensaje de descubri-

miento recibido, la dirección IP propuesta para el cliente, la máscara de red y **el tiempo de arrendamiento de la dirección** IP (el tiempo durante el que la dirección IP será válida). Es habitual que el servidor defina un tiempo de arrendamiento de varias horas o días [Droms 2002].

- *Solicitud DHCP.* El cliente recién llegado seleccionará de entre las ofertas de servidor y responderá a la oferta seleccionada con un **mensaje de solicitud DHCP**, devolviendo los parámetros de configuración.
- *ACK DHCP.* El servidor contesta al mensaje de solicitud DHCP con un **mensaje ACK DHCP**, que confirma los parámetros solicitados.

Una vez que el cliente recibe el mensaje de reconocimiento (ACK) DHCP, la interacción se completa y el cliente puede utilizar la dirección IP asignada por DHCP durante el tiempo de arrendamiento. Dado que un cliente puede desear utilizar su dirección durante más tiempo del arrendado, DHCP también proporciona un mecanismo que permite a un cliente renovar su tiempo de arrendamiento de una dirección IP.

El valor de la capacidad plug-and-play de DHCP es claro, considerando el hecho de que una alternativa sería configurar manualmente la dirección IP de un host. Recuerde al estudiante que va de la clase a la biblioteca y luego a su dormitorio con el portátil, uniéndose a una nueva subred y que obtiene así una nueva dirección IP en cada ubicación. Sería inimaginable que un administrador de sistemas tuviera que reconfigurar las computadoras portátiles en cada ubicación, y pocos estudiantes (excepto aquellos que asisten a clases de redes) son lo suficientemente expertos como para poder configurar manualmente sus portátiles. Sin embargo, en lo que respecta a la movilidad, DHCP presenta también algunas deficiencias. Puesto que se obtiene una nueva dirección IP de DHCP cada vez que un nodo se conecta a una nueva subred, una conexión TCP con una aplicación remota no podría mantenerse como un nodo móvil entre subredes. En el Capítulo 6 examinaremos la infraestructura de IP móvil, una extensión reciente de la infraestructura IP que permite a un nodo móvil utilizar una dirección permanente según se va desplazado entre subredes. Puede encontrar información adicional sobre DHCP en [Droms 2002] y [dhc 2009]. En Internet Systems Consortium [ISC 2009] hay disponible una implementación de referencia de código fuente abierto para DHCP.

Traducción de direcciones de red (NAT)

Después de haber estudiado las direcciones de Internet y el formato de los datagramas IPv4, somos completamente conscientes de que todo dispositivo IP necesita una dirección IP. Con la proliferación de las subredes domésticas y de oficina pequeña (SOHO, *Small Office, Home Office*), podría parecer que esto implica que, cuando una red SOHO desea instalar una LAN para conectar varias máquinas, el ISP debería asignar un rango de direcciones para cubrir todas las máquinas de la red SOHO. Si la subred creciera (por ejemplo, porque los niños en casa no sólo disponen ya de sus propias computadoras, sino que también tienen dispositivos PDA, teléfonos IP y Game Boys conectadas en red), habría que asignar un bloque de direcciones enorme. Pero ¿qué ocurre si el ISP ya ha asignado las porciones adyacentes al rango de direcciones actualmente en uso de la red SOHO? ¿Y qué persona normal querría (o necesitaría) saber cómo gestionar las direcciones IP de la red de su casa? Afortunadamente, existe una forma más simple de asignar direcciones que ha encontrado un uso cada

vez más amplio en escenarios de este tipo: la **traducción de direcciones de red (NAT, *Network Address Translation*)** [RFC 2663; RFC 3022].

La Figura 4.22 muestra el funcionamiento de un router NAT. Este router, que se encuentra en una vivienda, tiene una interfaz que es parte de la red doméstica situada en la parte derecha de la Figura 4.22. El direccionamiento dentro de la red doméstica es exactamente como hemos visto anteriormente (las cuatro interfaces de la red tienen la misma dirección de subred 10.0.0/24). El espacio de direcciones 10.0.0.0/8 corresponde a una de las tres partes del espacio de direcciones IP que está reservado en [RFC 1918] para una red privada o para un **ámbito** con direcciones privadas, como la red doméstica de la Figura 4.22. Un *ámbito con direcciones privadas* hace referencia a una red cuyas direcciones sólo tienen significado para los dispositivos internos de dicha red. Veamos por qué esto es importante. Considere el hecho de que existen cientos de miles de redes domésticas y que muchas utilizan el mismo espacio de direcciones, 10.0.0.0/24. Los dispositivos de una red doméstica dada pueden enviarse paquetes entre sí utilizando el direccionamiento 10.0.0.0/24. Sin embargo, los paquetes reenviados *hacia fuera* de la red doméstica, hacia Internet, evidentemente no pueden utilizar estas direcciones (ni como dirección de origen ni como dirección de destino), porque existen cientos de miles de redes que emplean ese bloque de direcciones. Es decir, las direcciones 10.0.0.0/24 sólo tienen significado dentro de una red doméstica dada. Pero si las direcciones privadas sólo tienen significado dentro de la red, ¿cómo se direccionan los paquetes cuando se envían a Internet o se reciben de Internet, donde necesariamente las direcciones tienen que ser únicas? Para entender esto hay que comprender cómo funciona NAT.

El router NAT *no parece* un router a ojos del mundo exterior. En su lugar, el router NAT se comporta de cara al exterior como un *único* dispositivo con una dirección IP *única*. En la Figura 4.22, todo el tráfico que sale del router doméstico hacia Internet tiene una dirección IP de origen igual a 138.76.29.7, y todo el tráfico que entra en él tienen que tener la dirección de destino 138.76.29.7. En resumen, el router NAT oculta los detalles de la red doméstica al mundo exterior. (Como nota al margen, posiblemente se esté preguntando dónde obtienen las computadoras de la red doméstica sus direcciones y dónde obtiene el router su dirección IP única. A menudo, la respuesta a ambas preguntas es la misma: ¡DHCP! El router obtiene su dirección del servidor DHCP del ISP y el router ejecuta un servidor DHCP para proporcionar direcciones a las computadoras, dentro del espacio de direcciones de la red doméstica controlada por el router NAT-DHCP.)

Si todos los datagramas que llegan al router NAT procedentes de la WAN tienen la misma dirección IP de destino (específicamente, la de la interfaz WAN del router NAT), entonces ¿cómo sabe el router a qué host interno debería reenviar un datagrama dado? El truco consiste en utilizar una **tabla de traducciones NAT** almacenada en el router NAT, e incluir los números de puerto, así como las direcciones IP en las entradas de la tabla.

Considere el ejemplo de la Figura 4.22. Suponga que un usuario de una red doméstica que utiliza el host con la dirección 10.0.0.1 solicita una página web almacenada en un servidor web (puerto 80) con la dirección IP 128.119.40.186. El host 10.0.0.1 asigna el número de puerto de origen (arbitrario) 3345 y envía el datagrama a la LAN. El router NAT recibe el datagrama, genera un nuevo número de puerto de origen, 5001, para el datagrama, sustituye la dirección IP de origen por su dirección IP de la red WAN 138.76.29.7, y sustituye el número de puerto de origen original 3345 por el nuevo número de puerto de origen 5001. Al generar un nuevo número de puerto de origen, el router NAT puede seleccionar cualquier número de puerto de origen que actualmente no se encuentre en la tabla de tra-

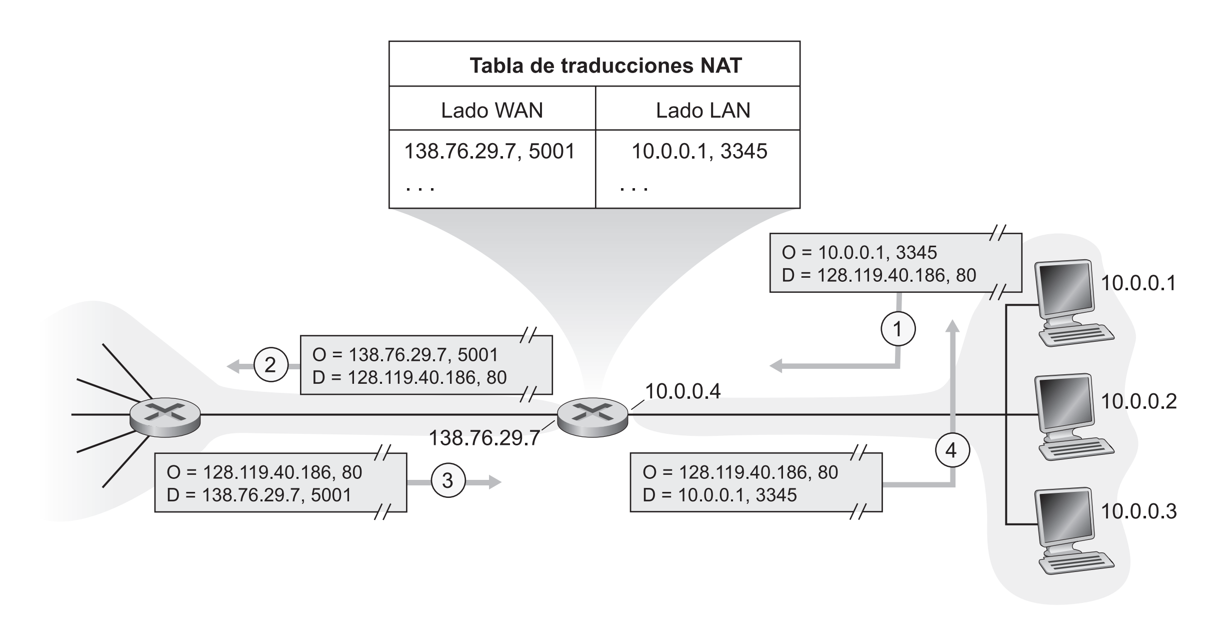


Figura 4.22 • Traducción de direcciones de red.

ducciones NAT. (Observe que, puesto que la longitud del campo número de puerto es 16 bits, el protocolo NAT puede dar soporte a 60.000 conexiones simultáneas utilizando la única dirección IP WAN del router.) En el router, NAT también añade una entrada a su tabla de traducciones. El servidor web, que afortunadamente no es consciente de que el datagrama entrante que contiene la solicitud HTTP ha sido manipulado por el router NAT, responde con un datagrama cuya dirección de destino es la dirección IP del router NAT y cuyo número de puerto de destino es 5001. Cuando este datagrama llega al router NAT, éste indexa la tabla de traducciones NAT utilizando la dirección IP de destino y el número de puerto de destino para obtener la dirección IP (10.0.0.1) y el número de puerto de destino (3345) apropiados para el navegador de la red doméstica. A continuación, el router reescribe la dirección de destino y el número de puerto de destino del datagrama y lo reenvía a la red doméstica.

NAT ha disfrutado de una gran difusión en los últimos años, aunque también tenemos que decir que muchos puristas de la comunidad IETF ponen bastantes objeciones a NAT. En primer lugar, argumentan que los números de puerto deben emplearse para direccionar procesos, no para direccionar hosts. (De hecho, esta violación puede causar problemas en los servidores en ejecución en la red doméstica, ya que, como hemos visto en el Capítulo 2, los procesos de servidor están a la espera de las solicitudes entrantes en los puertos bien conocidos.) En segundo lugar, argumentan que los routers están pensados para procesar paquetes sólo hasta la capa 3. En tercer lugar, opinan que el protocolo NAT viola lo que se ha venido a denominar el enfoque terminal a terminal, es decir, que los hosts deben comunicarse directamente entre sí, sin que los nodos intermedios modifiquen las direcciones IP y los números de puerto. Y en cuarto lugar, creen que debería utilizarse IPv6 (véase la Sección 4.4.4) para resolver la carestía de direcciones IP, en lugar de parchear el problema con una solución milagrosa como NAT. Pero, guste o no, NAT se ha convertido en un componente importante de Internet.

Otro problema más importante de NAT es que interfiere con las aplicaciones P2P, incluyendo las aplicaciones de compartición de archivos P2P y las aplicaciones de voz sobre IP P2P. Recuerde del Capítulo 2 que, en una aplicación P2P, cualquier par A participante debería poder iniciar una conexión TCP con cualquier otro par B participante. La esencia del problema es que si el par B está situado detrás de un traductor NAT no puede actuar como un servidor y aceptar conexiones TCP. Como veremos en los problemas de repaso, este problema de NAT puede soslayarse si el par A no está situado detrás de un traductor NAT. En este caso, el par A puede contactar primero al par B a través de un par C intermedio, que no esté situado detrás de un traductor NAT y con el que B tenga establecida una conexión TCP activa. El par A puede entonces pedir al par B, a través de C, que inicie una conexión TCP directamente con el par A. Una vez que la conexión TCP directa P2P se ha establecido entre los pares A y B, éstos podrán intercambiar mensajes o archivos. Esta técnica, conocida con el nombre de **inversión de la conexión (*connection reversal*)**, es utilizada por muchas aplicaciones P2P para **NAT transversal (NAT *Traversal*)**. Si tanto el par A como el par B utilizan NAT, la situación es un poco más compleja, pero puede solventarse utilizando retransmisores de aplicación, como hemos visto con los retransmisores Skype en el Capítulo 2.

UPnP

El mecanismo de NAT transversal es proporcionado cada vez más frecuentemente por Universal Plug and Play (UPnP), que es un protocolo que permite a un host descubrir y configurar un traductor NAT próximo [UPnP Forum 2009]. UPnP requiere que tanto el host como el traductor NAT sean compatibles con UPnP. Con UPnP, una aplicación que se ejecuta en un host puede solicitar una correspondencia NAT entre su tupla (*dirección IP privada, número de puerto privado*) y la tupla (*dirección IP pública, número de puerto público*) para algún número de puerto público solicitado. Si el traductor NAT acepta la solicitud y crea la correspondencia, entonces los nodos del exterior pueden iniciar conexiones TCP con (*dirección IP privada, número de puerto privado*). Además, UPnP permite a la aplicación conocer el valor de (*dirección IP pública, número de puerto público*), de manera que la aplicación pueda anunciarse al mundo exterior.

Por ejemplo, suponga que su host, situado detrás de un traductor NAT compatible con UPnP, tiene la dirección privada 10.0.0.1 y está ejecutando BitTorrent en el puerto 3345. Suponga también que la dirección IP pública del traductor NAT es 138.76.29.7. Naturalmente, su aplicación BitTorrent desea poder aceptar conexiones de otros hosts, para poder intercambiar fragmentos con ellos. En esta situación, la aplicación BitTorrent de su host pide al traductor NAT que cree un "túnel" que asigne (10.0.0.1, 3345) a (138.76.29.7, 5001). (La aplicación elige el número de puerto público 5001.) La aplicación BitTorrent de su host podría también anunciar a su tracker que está disponible en (138.76.29.7, 5001). De esta manera, un host externo que esté ejecutando BitTorrent puede contactar con el tracker y determinar que su aplicación BitTorrent está ejecutándose en (138.76.29.7, 5001). El host externo puede enviar un paquete SYN TCP a (138.76.29.7, 5001). Cuando el traductor NAT recibe el paquete SYN, cambiará la dirección IP de destino y el número de puerto del paquete a (10.0.0.1, 3345) y lo reenviará a través de NAT.

En resumen, UPnP permite a hosts externos iniciar sesiones de comunicación con hosts conectados a través de un traductor NAT utilizando TCP o UDP. NAT ha sido durante mucho tiempo un auténtico quebradero de cabeza para las aplicaciones P2P; UPnP, que proporciona una solución efectiva y robusta de NAT transversal, puede ser su salvador. Esta exposición acerca de NAT y UPnP ha sido necesariamente breve, por lo que si desea saber más acerca de NAT, consulte [Huston 2004, Cisco NAT 2009].

4.4.3 Protocolo de mensajes de control de Internet (ICMP)

Recuerde que la capa de red de Internet tiene tres componentes principales: el protocolo IP, estudiado en la sección anterior; los protocolos de enrutamiento de Internet (incluyendo RIP, OSPF y BGP), que se abordan en la Sección 4.6 e ICMP, que es el objeto de esta sección.

Los hosts y los routers utilizan ICMP, especificado en [RFC 792], para intercambiarse información acerca de la capa de red. El uso más típico de ICMP es la generación de informes de error. Por ejemplo, al ejecutar una sesión Telnet, FTP o HTTP, puede encontrarse con un mensaje de error como "Red de destino inalcanzable". Este mensaje tiene su origen en ICMP. En algún momento, un router IP no ha podido encontrar una ruta hasta el host especificado en su aplicación Telnet, FTP o HTTP, y dicho router ha creado y enviado un mensaje ICMP de tipo 3 a su host para informarle del error.

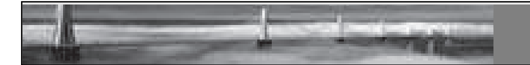
ICMP a menudo se considera parte de IP pero, en sentido arquitectónico, se encuentra justo encima de IP, ya que los mensajes ICMP son transportados dentro de los datagramas IP. Es decir, los mensajes ICMP son transportados como carga útil de IP, al igual que los segmentos TCP o UDP son transportados como carga útil de IP. De forma similar, cuando un host recibe un datagrama IP con ICMP especificado como el protocolo de la capa superior, demultiplexa el contenido del datagrama para ICMP, al igual que demultiplexaría el contenido de un datagrama para TCP o UDP.

Los mensajes ICMP tienen un campo de tipo y un campo de código, y contienen la cabecera y los 8 primeros bytes del datagrama IP que ha dado lugar a la generación del mensaje ICMP en primer lugar (de modo que el emisor puede determinar qué datagrama ha producido el error). En la Figura 4.23 se muestran una serie de tipos de mensajes seleccionados. Observe que los mensajes ICMP no sólo se emplean para indicar condiciones de error.

Tipo ICMP	Código	Descripción
0	0	respuesta de eco (para ping)
3	0	red de destino inalcanzable
3	1	host de destino inalcanzable
3	2	protocolo de destino inalcanzable
3	3	puerto de destino inalcanzable
3	6	red de destino desconocida
3	7	host de destino desconocido
4	0	regulación del origen (control de congestión)
8	0	solicitud de eco
9	0	anuncio de router
10	0	descubrimiento de router
11	0	TTL caducado
12	0	Cabecera IP errónea

Figura 4.23 • Tipos de mensajes ICMP.

El programa bien conocido `ping` envía un mensaje ICMP de tipo 8 y código 0 al host especificado. El host de destino, al ver la solicitud de eco, devuelve una respuesta de eco ICMP de tipo 0 y código 0. La mayoría de las implementaciones de TCP/IP soportan el servidor `ping` directamente en el sistema operativo; es decir, el servidor no es un proceso. El Capítulo 11 de [Stevens 1990] proporciona el código fuente del programa cliente `ping`.



SEGURIDAD

INSPECCIÓN DE DATAGRAMAS: CORTAFUEGOS Y SISTEMAS DE DETECCIÓN DE INTRUSIONES

Suponga que le han asignado la tarea de administrar una red doméstica, departamental, universataria o corporativa. Los atacantes, que saben cuál es el rango de direcciones IP de su red, pueden enviar fácilmente datagramas IP a las direcciones de ese rango. Estos datagramas pueden hacer toda clase de cosas retorcidas, incluyendo confeccionar mapas de la red mediante barridos de ping y escaneo de puertos, dañar los hosts vulnerables con paquetes erróneos, inundar los servidores con una enrome cantidad de paquetes ICMP e infectar los hosts incluyendo software malicioso en los paquetes. Como administrador de la red, ¿qué hará con todos esos malvados capaces de enviar paquetes maliciosos a su red? Hay disponibles dos mecanismos de defensa muy populares contra los ataques de paquetes maliciosos: los cortafuegos y los sistemas de detección de intrusiones (IDS, *Intrusion Detection System*).

Como administrador de la red, en primer lugar puede probar a instalar un cortafuegos entre su red e Internet. (La mayoría de los routers actuales de acceso disponen de cortafuegos.) Los cortafuegos inspeccionan los campos de cabecera de los segmentos y datagramas, denegando el acceso a la red interna a los datagramas sospechosos. Por ejemplo, un cortafuegos puede configurarse para bloquear todos los paquetes de solicitud de eco ICMP, impidiendo así que un atacante lleve a cabo un tradicional barrido de ping a lo largo de su rango de direcciones IP. Los cortafuegos también pueden bloquear paquetes basándose en las direcciones IP de origen y de destino y en los números de puerto. Además, los cortafuegos se pueden configurar para controlar las conexiones TCP, dejando entrar sólo a los datagramas que pertenecen a conexiones aprobadas.

Con un sistema IDS puede proporcionarse protección adicional. Un IDS, colocado normalmente en la frontera de la red, realiza una "inspección profunda de los paquetes", examinando no sólo los campos de cabecera sino también las cargas útiles de los datagramas (incluyendo los datos de la capa de aplicación). Un IDS dispone de una base de datos de signaturas de paquete que se sabe que forman parte de ataques. Esta base de datos se actualiza automáticamente cuando se descubren nuevos tipos de ataque. A medida que los paquetes atraviesan el sistema IDS, éste intenta encontrar una coincidencia de los campos de cabecera o las cargas útiles con las signaturas que almacena en su base de datos. Si encuentra una coincidencia, genera una alerta. Un sistema de prevención de intrusiones (IPS, *Intrusion Prevention System*) es similar a un sistema IDS, salvo porque además de generar una alerta bloquea los paquetes. En el Capítulo 8 estudiaremos los cortafuegos y los sistemas IDS en detalle.

¿Pueden los cortafuegos y los sistemas IDS proteger completamente a la red frente a todos los ataques? Evidentemente, la respuesta es no, ya que los atacantes encuentran continuamente nuevos ataques para los que las signaturas todavía no están disponibles. No obstante, los cortafuegos y los IDS tradicionales basados en signaturas son útiles para proteger a las redes de los ataques conocidos.

Observe que el programa cliente necesita poder instruir al sistema operativo para generar un mensaje ICMP de tipo 8 y código 0.

Otro interesante mensaje ICMP es el mensaje de regulación del origen. Este mensaje rara vez se emplea en la práctica. Su propósito original era llevar a cabo el control de congestión (permitir a un router congestionado enviar un mensaje ICMP de este tipo a un host para forzarle a reducir su velocidad de transmisión). Hemos visto en el Capítulo 3 que TCP dispone de su propio mecanismo control de congestión que opera en la capa de transporte, sin utilizar ninguna realimentación de la capa de red, como el mensaje ICMP de regulación del origen.

En el Capítulo 1 hemos introducido el programa `Traceroute`, el cual nos permite trazar una ruta desde un host a cualquier otro host del mundo. Cada vez más frecuentemente, Traceroute se implementa con mensajes ICMP. Para determinar los nombres y las direcciones de los routers existentes entre el origen y el destino, `Traceroute` en el origen envía una serie de datagramas IP ordinarios al destino. Cada uno de estos datagramas transporta un segmento UDP con un número de puerto UDP poco probable. El primero de estos datagramas tiene un TTL de 1, el segundo de 2, el tercero de 3, y así sucesivamente. El origen también inicia los temporizadores para cada uno de los datagramas. Cuando el datagrama *n-ésimo* llega al router *n-ésimo*, éste observa que el TTL del datagrama acaba de caducar. De acuerdo con las reglas del protocolo IP, el router descarta el datagrama y envía al origen un mensaje de advertencia ICMP (tipo 11, código 0). Este mensaje de advertencia incluye el nombre del router y su dirección IP. Cuando este mensaje ICMP llega de vuelta al origen, éste obtiene el tiempo de ida y vuelta del temporizador, y el nombre y la dirección IP del router *n-ésimo* del propio mensaje ICMP.

¿Cómo sabe un origen `Traceroute` cuándo dejar de enviar segmentos UDP? Recuerde que el origen incrementa el valor del campo TTL cada vez que envía un datagrama. Por tanto, uno de los datagramas terminará recorriendo el camino completo hasta el host de destino. Dado que ese datagrama contiene un segmento UDP con un número de puerto improbable, el host de destino devuelve al origen un mensaje ICMP de puerto inalcanzable (tipo 3, código 3). Cuando el host de origen recibe este mensaje ICMP, sabe que no tiene que enviar más paquetes de prueba. (Realmente, el programa estándar `Traceroute` envía conjuntos de tres paquetes con el mismo TTL y, por tanto, la salida de `Traceroute` proporciona tres resultados para cada TTL.)

De esta forma, el host de origen obtiene el número y la identidad de los routers que existen entre él y el host de destino, así como el tiempo de ida y vuelta entre los dos hosts. Observe que el programa cliente `Traceroute` tiene que poder instruir al sistema operativo para generar los datagramas UDP con valores TTL específicos y el sistema operativo también tiene que ser capaz de notificarle la llegada de mensajes ICMP. Ahora que sabe cómo funciona `Traceroute`, puede volver atrás y prácticar con él un poco más.

4.4.4 IPv6

A principios de la década de 1990, el *Internet Engineering Task Force* comenzó a desarrollar un sucesor para el protocolo IPv4. La principal motivación de esta iniciativa fue que se dieron cuenta de que el espacio de direcciones IP de 32 bits estaba comenzando a agotarse, a causa de las nuevas subredes y nodos IP que estaban conectándose a Internet (a los que se les estaban asignando direcciones IP únicas) a una velocidad sobrecogedora. Para responder a esta necesidad de un espacio de direcciones IP más grande, se desarrolló un nuevo proto-

colo IP, el protocolo IPv6. Los diseñadores de IPv6 también vieron aquí la oportunidad de ajustar y aumentar otros aspectos de IPv4, basándose en la experiencia acumulada sobre el funcionamiento de IPv4.

El momento en que todas las direcciones IPv4 habrían sido asignadas (y por tanto ninguna nueva subred podría conectarse a Internet) fue objeto de un importante debate. Las estimaciones de los dos líderes del grupo de trabajo *Address Lifetime Expectations* (Expectativas del tiempo de vida de las direcciones) del IETF fueron que las direcciones comenzarían a agotarse en 2008 y 2018, respectivamente [Solensky 1996]. Un análisis más reciente [Huston 2008] sitúa la fecha de agotamiento en torno a 2010. En 1996, el Registro americano de números de Internet (ARIN, *American Registry for Internet Numbers*) informó de que todas las direcciones de clase A IPv4 habían sido asignadas, el 62 por ciento de las direcciones de clase B habían sido asignadas y el 37 por ciento de las direcciones de clase C también habían sido asignadas [ARIN 1996]. Si desea ver un informe reciente sobre la asignación del espacio de direcciones IPv4, consulte [Hain 2005]. Aunque estas estimaciones y datos sugerían que quedaba bastante tiempo para que el espacio de direcciones de IPv4 se agotara, se dieron cuenta de que se necesitaría un tiempo considerable para implantar una nueva tecnología a tan gran escala, y por eso se comenzó a trabajar en IPng (*Next Generation* IP, IP de siguiente generación) [Bradner 1996; RFC 1752]. El resultado de este esfuerzo fue la especificación de la versión 6 de IP (IPv6) [RFC 2460]. (Una pregunta que se plantea a menudo es qué ocurrió con IPv5. Inicialmente se pensó que el protocolo ST-2 se convertiría en IPv5, pero ST-2 fue descartado más tarde en favor del protocolo RSVP, que estudiaremos en el Capítulo 7.)

Puede encontrar fuentes de información excelentes acerca de IPv6 en [Huitema 1998, IPv6 2009].

Fomato del datagrama IPv6

El formato del datagrama IPv6 se muestra en la Figura 4.24. Los cambios más importantes introducidos en IPv6 son evidentes en el formato de su datagrama:

- *Capacidades ampliadas de direccionamiento.* IPv6 aumenta el tamaño de la dirección IP de 32 a 128 bits. De esta manera, se asegura que el mundo no se quedará sin direccio-

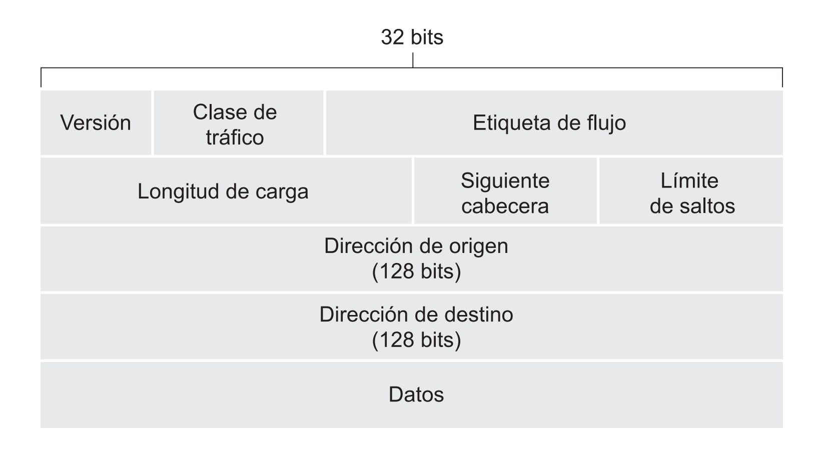


Figura 4.24 • Formato del datagrama de IPv6.

nes IP. Ahora, cada grano de arena del planeta puede tener asignada una dirección IP. Además de las direcciones de unidifusión y de multidifusión, IPv6 ha introducido un nuevo tipo de dirección, denominado **dirección anycast**, que permite entregar un datagrama a uno cualquiera de un grupo de hosts. (Esta funcionalidad podría utilizarse, por ejemplo, para enviar un mensaje GET HTTP al más próximo de una serie de sitios espejo que contengan un determinado documento.)

- *Una cabecera de 40 bytes simplificada.* Como se ha mencionado anteriormente, algunos de los campos de IPv4 se han eliminado o se han hecho opcionales. La cabecera de longitud fija de 40 bytes permite un procesamiento más rápido del datagrama IP. Una nueva codificación de las opciones permite un procesamiento más flexible de las mismas.
- *Prioridad y etiquetado del flujo.* IPv6 utiliza una definición bastante amplia de **flujo**. Los documentos RFC 1752 y RFC 2460 establecen que esto permite "etiquetar los paquetes que pertenecen a determinados flujos para los que el emisor solicita un tratamiento especial, como un servicio en tiempo real o una calidad de servicio no predeterminados". Por ejemplo, la transmisión de audio y de vídeo puede posiblemente tratarse como un flujo. Por el contrario, aplicaciones más tradicionales, como la transferencia de archivos y el correo electrónico, no se pueden tratar como flujos. Es posible que el tráfico transportado por un usuario de alta prioridad (por ejemplo, alguien que paga por obtener un mejor servicio para su tráfico) tenga que ser tratado como un flujo. Sin embargo, lo que está claro es que los diseñadores de IPv6 prevén la eventual necesidad de poder diferenciar entre los flujos, incluso aunque el significado exacto de flujo no haya sido determinado. La cabecera de IPv6 también tiene un campo de 8 bits para definir la clase de tráfico. Este campo, como el campo TOS de IPv4, se puede utilizar para dar prioridad a determinados datagramas de un flujo, o se puede emplear para dar prioridad a datagramas de ciertas aplicaciones (por ejemplo, ICMP) con respecto a datagramas de otras aplicaciones (como por ejemplo, las noticias de red).

Como se ha mencionado anteriormente, la comparación de la Figura 4.24 con la Figura 4.13 revela la estructura más simple y estilizada del datagrama de IPv6. En IPv6 se definen los siguientes campos:

- *Versión.* Este campo de 4 bits identifica el número de versión IP. Nada sorprendentemente, IPv6 transporta un valor de 6 en este campo. Observe que incluir en este campo el valor 4 no implica que se cree un datagrama IPv4 válido. Si lo hiciera, la vida sería mucho más simple (véase más adelante la exposición sobre la transición de IPv4 a IPv6).
- *Clase de tráfico.* Este campo de 8 bits es similar en espíritu al campo TOS que hemos visto en IPv4.
- *Etiqueta de flujo.* Como hemos mencionado anteriormente, este campo de 20 bits se utiliza para identificar un flujo de datagramas.
- *Longitud de la carga útil.* Este valor de 16 bits se trata como un entero sin signo que proporciona el número de bytes del datagrama IPv6 incluidos a continuación de la cabecera del datagrama, que es de 40 bytes y tiene longitud fija.
- *Siguiente cabecera.* Este campo identifica el protocolo al que se entregará el contenido (el campo de datos) de este datagrama (por ejemplo, a TCP o UDP). El campo utiliza los mismos valores que el campo de protocolo de la cabecera de IPv4.

- *Límite de saltos.* Cada router que reenvía un datagrama decrementa el contenido de este campo en una unidad. Si el límite de saltos alcanza el valor cero, el datagrama se descarta.
- *Direcciones de origen y de destino.* Los distintos formatos de la dirección de 128 bits de IPv6 se describen en el documento RFC 4291.
- *Datos.* Ésta es la parte de la carga útil del datagrama de IPv6. Cuando el datagrama llegue a su destino, la carga útil se extraerá del datagrama IP y se pasará al protocolo especificado en el campo Siguiente cabecera.

Hasta aquí hemos identificado el propósito de los campos incluidos en los datagramas de IPv6. Comparando el formato del datagrama de IPv6 de la Figura 4.24 con el formato del datagrama de IPv4 que hemos visto en la Figura 4.13, podemos observar que varios campos del datagrama de IPv4 ya no aparecen en IPv6:

- *Fragmentación/Reensamblado.* IPv6 no permite ni la fragmentación ni el reensamblado en routers intermedios; estas operaciones sólo pueden ser realizadas por el origen y el destino. Si un router recibe un datagrama IPv6 y es demasiado largo para ser reenviado por el enlace de salida, el router simplemente lo descarta y envía de vuelta al emisor un mensaje de error ICMP "Paquete demasiado grande" (véase más adelante). El emisor puede entonces reenviar los datos utilizando un tamaño de datagrama IP más pequeño. La fragmentación y el reensamblado son operaciones que consumen tiempo, por lo que eliminando esta funcionalidad de los routers e incluyéndola directamente en los sistemas terminales se acelera considerablemente el reenvío IP dentro de la red.
- *Suma de comprobación de cabecera.* Puesto que los protocolos de la capa de transporte (por ejemplo, TCP y UDP) y de la capa de enlace de datos (por ejemplo, Ethernet) en las capas de Internet realizan sumas de comprobación, los diseñadores de IP probablemente pensaron que esta funcionalidad ya era suficientemente redundante en la capa de red y podía eliminarse. Una vez más, el procesamiento rápido de los paquetes IP era la preocupación principal. Recuerde que en la Sección 4.4.1 hemos visto que dado que la cabecera de IPv4 contiene un campo TTL (similar al campo límite de saltos de IPv6), la suma de comprobación de la cabecera de IPv4 necesitaba ser recalculada en cada router. Al igual que la fragmentación y el reensamblado, ésta también era una operación muy costosa en IPv4.
- *Opciones.* La cabecera IP estándar ya no incluye un campo de opciones. Sin embargo, las opciones no han desaparecido. En su lugar, el campo de opciones es una de las posibles siguientes cabeceras apuntadas desde dentro de la cabecera IPv6. Es decir, al igual que las cabeceras de los protocolos TCP o UDP pueden ser la siguiente cabecera dentro de un paquete IP, también puede serlo un campo de opciones. La eliminación del campo de opciones da como resultado una cabecera IP de 40 bytes de longitud fija.

Recuerde que en la Sección 4.4.3 hemos visto que los nodos IP utilizan el protocolo ICMP para informar de condiciones de error y proporcionar información limitada (por ejemplo, la respuesta de eco de un mensaje `ping`) a un sistema terminal. En el documento RFC 4443 se ha definido una nueva versión de ICMP para IPv6. Además de reorganizar las definiciones de tipos y códigos ICMP existentes, ICMPv6 también añadió nuevos tipos y códigos requeridos por la nueva funcionalidad de IPv6, entre los que se incluyen el tipo "Paquete demasiado grande" y el código de error "Opciones IPv6 no reconocidas". Además, ICMPv6

incluye la funcionalidad del Protocolo de gestión de grupos de Internet (IGMP, *Internet Group Management Protocol*) que estudiaremos en la Sección 4.7. IGMP, que se emplea para gestionar el modo en que un host se une a un grupo de multidifusión y lo abandona, anteriormente era un protocolo separado de ICMP en IPv4.

Transición de IPv4 a IPv6

Ahora que hemos visto los detalles técnicos de IPv6, vamos a considerar una cuestión práctica: ¿cómo va a hacer Internet, que está basada en IPv4, la transición a IPv6? El problema está en que mientras que los nuevos sistemas para IPv6 pueden ser compatibles en sentido descendente, es decir, pueden enviar, enrutar y recibir datagramas IPv4, los sistemas para IPv4 ya implantados no son capaces de manejar datagramas de IPv6. Existen varias posibles soluciones.

Una opción sería declarar un día D (una fecha y hora en la que todas las máquinas conectadas a Internet se apagaran y actualizaran de IPv4 a IPv6). La última transición importante de una tecnología a otra (de NCP a TCP como servicio de transporte fiable) se hizo hace casi 25 años. Incluso entonces [RFC 801], cuando Internet era pequeña y todavía era administrada por un número pequeño de "magos", se dieron cuenta de que establecer un día D no era posible, por lo que un día D que implique a cientos de millones de máquinas y a millones de administradores y usuarios de red es aún más impensable hoy día. El documento RFC 4213 describe dos métodos (que pueden utilizarse juntos o por separado) para integrar de forma gradual los hosts y routers IPv6 en el mundo de IPv4 (por supuesto, con el objetivo a largo plazo de que todos los nodos IPv4 hagan la transición a IPv6).

Probablemente, la forma más directa de introducir nodos IPv6 es mediante el método de **pila dual**, en el que los nodos IPv6 también disponen de una implementación IPv4 completa. Un nodo así, conocido como nodo IPv6/IPv4 en el RFC 4213, tiene la capacidad de enviar y recibir tanto datagramas IPv4 como IPv6. Al interoperar con un nodo IPv4, un nodo IPv6/IPv4 puede utilizar datagramas IPv4; y al interoperar con un nodo IPv6, puede hablar en IPv6. Los nodos IPv6/IPv4 tienen que tener direcciones IPv6 e IPv4. Además, tienen que ser capaces de determinar si otro nodo es un nodo IPv6 o sólo IPv4. Este problema puede resolverse utilizando DNS (véase el Capítulo 2), que puede devolver una dirección IPv6 si el nombre del nodo que se está resolviendo es IPv6 o, en cualquier otro caso, devolver una dirección IPv4. Por supuesto, si el nodo que envía la solicitud DNS es un nodo exclusivamente IPv4, DNS devuelve una dirección IPv4.

En el método de pila dual, si el emisor o el receptor son nodos exclusivamente IPv4 tiene que utilizarse un datagrama IPv4. Como resultado, es posible que dos nodos IPv6 puedan terminar enviándose entre ellos datagramas IPv4. Esto se ilustra en la Figura 4.25. Suponga que el nodo A es un nodo IPv6 y desea enviar un datagrama IP al nodo F, que también es un nodo IPv6. Los nodos A y B pueden intercambiarse un datagrama IPv6. Sin embargo, el nodo B tiene que crear un datagrama IPv4 para enviárselo al nodo C. Realmente, el campo de datos del datagrama IPv6 se puede copiar en el campo de datos del datagrama IPv4 y puede establecer la correspondencia apropiada de direcciones. Sin embargo, al hacer la conversión de IPv6 a IPv4, habrá campos específicos de IPv6 en el datagrama IPv6 (por ejemplo, el campo identificador de flujo) que no tienen una contrapartida en IPv4. La información de estos campos se pierde. Por tanto, incluso aunque E y F puedan intercambiar datagramas IPv6, los datagramas IPv4 que llegan a E procedentes de D no contienen todos los campos que había en el datagrama IPv6 original enviado por A.

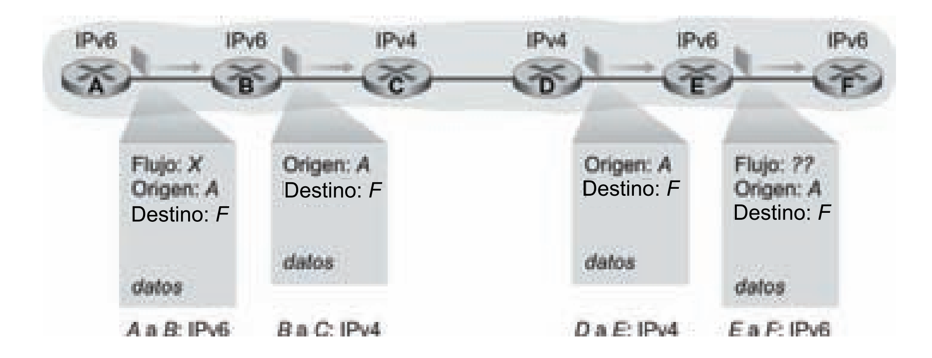


Figura 4.25 • Método de la pila dual.

Una alternativa al método de la pila dual, también incluida en el RFC 4213, se conoce como **tunelización**. La tunelización puede resolver el problema anteriormente mencionado permitiendo, por ejemplo, que E reciba los datagramas IPv6 originados en el nodo A. La idea básica en la que descansa el método de tunelización es la siguiente: suponga que dos nodos IPv6 (por ejemplo los nodos B y E de la Figura 4.25) desean interoperar utilizando datagramas IPv6 pero están conectados entre sí a través de routers IPv4. Nos referiremos al conjunto de routers intermedios IPv4 existentes entre los dos routers IPv6 como a un **túnel**, como se ilustra en la Figura 4.26. Mediante la tunelización, el nodo IPv6 del lado emisor del

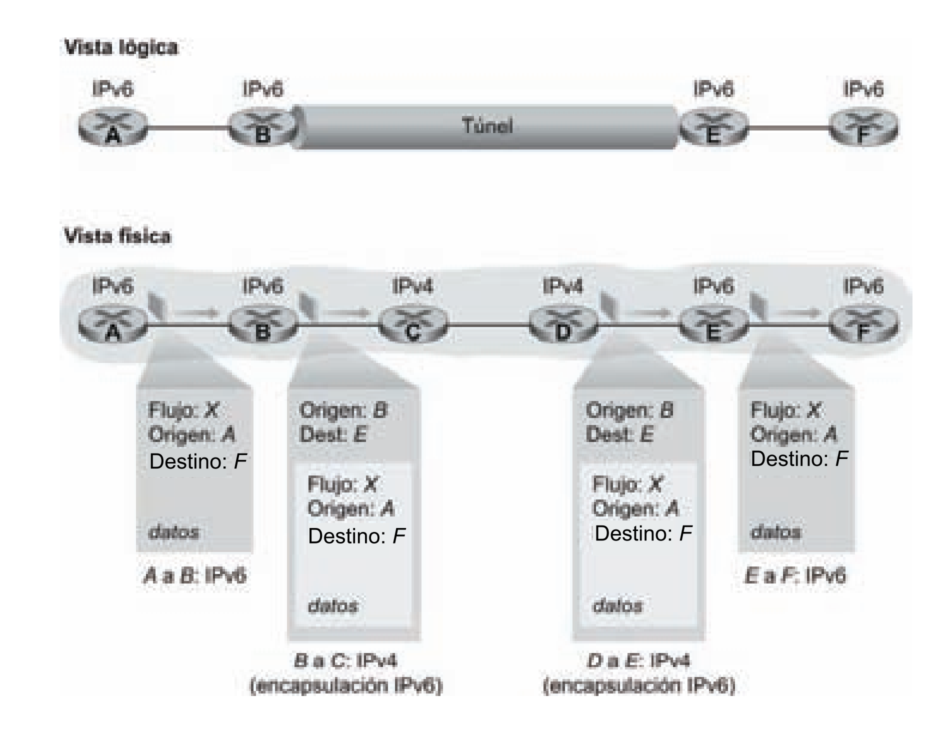


Figura 4.26 • Tunelización.

túnel (por ejemplo, B) toma el datagrama IPv6 *completo* y lo incluye en el campo de datos (carga útil) de un datagrama IPv4. Este datagrama IPv4 se direcciona entonces hacia el nodo IPv6 del lado receptor del túnel (por ejemplo, E) y se envía al primer nodo del túnel (por ejemplo, C). Los routers IPv4 existentes en el túnel enrutan este datagrama IPv4 entre ellos mismos, como si fuera cualquier otro datagrama, siendo totalmente inconscientes de que el datagrama IPv4 contiene un datagrama IPv6 completo. El nodo IPv6 del lado de recepción del túnel termina recibiendo el datagrama IPv4 (¡es el destino del datagrama IPv4!), determina que el datagrama IPv4 contiene un datagrama IPv6, extrae el datagrama IPv6 y, a continuación, le enruta exactamente igual que si hubiera recibido el datagrama IPv6 desde un vecino IPv6 directamente conectado.

Para terminar esta sección, diremos que aunque la adopción de IPv6 inicialmente fue lenta [Lawton 2001], recientemente está empezando a tomar impulso. Consulte [Huston 2008b] para obtener información acerca del grado de implantación de IPv6 en 2008. La *Office of Management and Budget* de Estados Unidos ha ordenado que los routers troncales de las redes del gobierno sean capaces de soportar IPv6 a mediados de 2008; algunas de las agencias gubernamentales han cumplido este mandato en el momento de escribir este libro (noviembre de 2008). La proliferación de dispositivos como los teléfonos IP y otros dispositivos portátiles proporciona más motivos para una implantación más extensa de IPv6. El programa europeo *Third Generation Partnership Program* [3GPP 2009] ha especificado IPv6 como el esquema de direccionamiento estándar para los dispositivos multimedia móviles. Aunque IPv6 no haya sido implantado ampliamente en los primeros 10 años de su corta vida, evidentemente hay que enjuiciar su éxito a largo plazo. El sistema de numeración telefónico actual tardó varias décadas en asentarse, pero lleva en uso casi medio siglo y sin signos de desaparecer. De forma similar, la adopción de IPv6 llevará su tiempo, pero también esta tecnología estará después mucho tiempo entre nosotros. Brian Carpenter, antiguo jefe del *Internet Architecture Board* [IAB 2009] y autor de varios RFC relativos a IPv6, dice: "Siempre he visto esto como un proceso de 15 años que comenzó en 1995" [Lawton 2001]. Por lo que, según las fechas de Carpenter, estamos al final del proceso inicial de implantación.

Una lección importante que podemos aprender de la experiencia de IPv6 es que es enormemente complicado cambiar los protocolos de la capa de red. Desde principios de la década de 1990, numerosos protocolos de la capa de red nuevos se han promulgado como la siguiente revolución más importante de Internet, pero la mayoría de estos procotolos han tenido una penetración limitada hasta la fecha. Entre estos protocolos se incluyen IPv6, los protocolos de mutlidifusión (Sección 4.7) y los protocolos de reserva de recursos (Capítulo 7). De hecho, la introducción de nuevos protocolos en la capa de red es como sustituir los cimientos de una casa (es difícil de hacer sin tirar abajo toda la casa o al menos reubicar temporalmente a sus habitantes). Por otro lado, Internet ha sido testigo de la rápida implantación de nuevos protocolos en la capa de aplicación. Ejemplos clásicos de esto son, por supuesto, la Web, la mensajería instantánea y la compartición de archivos P2P. Otros ejemplos son los flujos de audio y de vídeo y los juegos distribuidos. Introducir protocolos de la capa de aplicación nuevos es como añadir una nueva capa de pintura a las paredes de una casa (es relativamente fácil de hacer y, si se elige un color atractivo, otras personas del vecindario le copiarán). En resumen, en el futuro podemos esperar ver cambios en la capa de red de Internet, pero probablemente esos cambios se producirán en una escala temporal mucho más lenta que los cambios que se producirán en la capa de aplicación.

4.4.5 Una breve incursión en la seguridad IP

En la Sección 4.4.3 hemos visto en detalle IPv4, incluyendo los servicios que proporciona y cómo se implementan dichos servicios. Es posible que al leer la sección se haya dado cuenta de que no hemos mencionado los servicios de seguridad. De hecho, IPv4 fue diseñado en una época (en la década de 1970) en la que Internet era utilizada principalmente por investigadores dedicados a las redes que confiaban unos en otros. Crear una red de computadoras que integrara una multitud de tecnologías de la capa de enlace ya constituía un enorme reto, sin tener además que preocuparse por la seguridad.

Pero siendo actualmente la seguridad uno de los problemas más importantes, los investigadores de Internet han tenido que diseñar nuevos protocolos de la capa de red que proporcionen una amplia variedad de servicios de seguridad. Uno de estos protocolos es IPsec, uno de los protocolos de la capa de red seguros más populares y también ampliamente implementado en las redes privadas virtuales (VPN, *Virtual Private Network*). Aunque IPsec y sus fundamentos criptográficos se cubren con cierto detalle en el Capítulo 8, proporcionamos en esta sección una breve introducción de carácter general a los servicios IPsec.

IPsec ha sido diseñado para ser compatible hacia abajo con IPv4 e IPv6. En concreto, para recoger los beneficios de IPsec, no necesitamos sustituir las pilas de protocolos de *todos* los routers y hosts de Internet. Por ejemplo, en el modo transporte (uno de los dos “modos” de IPsec), si dos hosts desean comunicarse de forma segura, basta con que IPsec esté disponible en esos dos hosts. Todos los demás routers y hosts pueden continuar ejecutando IPv4 normal y corriente.

Concretando, aquí vamos a centrarnos en el modo transporte de IPsec. En este modo, en primer lugar, dos hosts establecen una sesión IPsec entre ellos (por tanto, IPsec es un protocolo orientado a la conexión). Una vez establecida la conexión, todos los segmentos TCP y UDP enviados entre los dos hosts disfrutan de los servicios de seguridad proporcionados por IPsec. En el lado emisor, la capa de transporte pasa un segmento a IPsec. A continuación, IPsec cifra el segmento, añade al segmento campos de seguridad adicionales y encapsula la carga útil resultante en un datagrama IP ordinario (realmente es algo más complicado que esto como veremos en el Capítulo 8.) El host emisor envía entonces el datagrama a Internet, que los transporta hasta su host de destino, donde IPsec descifra el segmento y pasa el segmento no cifrado a la capa de transporte.

Entre los servicios proporcionados en una sesión IPsec se incluyen:

- *Negociación criptográfica.* Mecanismos que permiten a los dos hosts que se están comunicando acordar las claves y algoritmos criptográficos.
- *Cifrado de la carga útil de los datagramas IP*. Cuando el host emisor recibe un segmento de la capa de transporte, IPsec cifra la carga útil. La carga útil sólo puede ser descifrada por IPsec en el host receptor.
- *Integridad de los datos*. IPsec permite al host receptor verificar que los campos de cabecera del datagrama y la carga útil cifrada no han sido modificados cuando el datagrama estaba en la ruta entre el origen y el destino.
- *Autenticación del origen*. Cuando un host recibe un datagrama IPsec de un origen de confianza (con una clave segura, véase el Capítulo 8), el host está seguro de que la dirección IP de origen del datagrama es el origen real del mismo.

Cuando dos hosts han establecido una sesión IPsec entre sí, todos los segmentos TCP y UDP enviados entre ellos son cifrados y autenticados. Por tanto, IPsec proporciona una

cobertura a todo riesgo, garantizando la seguridad de todas las comunicaciones entre los dos hosts para todas las aplicaciones de red.

Una empresa puede utilizar IPsec para comunicarse de forma segura a través de la red Internet pública no segura. Con propósitos ilustrativos, veamos ahora un ejemplo sencillo. Sea una empresa con una gran cantidad de personal comercial que viaja, y cada comercial posee una computadora portátil. Suponga que los comerciales necesitan consultar con frecuencia información confidencial de la empresa (por ejemplo, los precios y datos de los productos), que está almacenada en un servidor situado en la oficina central de la empresa. Suponga que los comerciales también necesitan enviar documentos confidenciales entre ellos. ¿Cómo puede hacerse esto utilizando IPsec? Como habrá imaginado, instalamos IPsec en el servidor y en todas las computadoras portátiles de los comerciales. Con IPsec instalado en estos hosts, cuando un comercial necesita comunicarse con el servidor o con otro comercial, la sesión de comunicación será segura.

4.5 Algoritmos de enrutamiento

Hasta aquí nos hemos ocupado principalmente de la función de reenvío de la capa de red. Hemos visto que cuando un paquete llega a un router, éste busca en la tabla de reenvío y determina la interfaz de enlace a la que tiene que dirigir el paquete. También hemos visto que los algoritmos de enrutamiento, que operan en los routers de red, intercambian y calculan la información que se utiliza para configurar estas tablas de reenvío. En la Figura 4.2 se ha mostrado la relación existente entre los algoritmos de enrutamiento y las tablas de reenvío. Una vez que hemos estudiado con cierta profundidad la función de reenvío, vamos a pasar al otro tema central de este capítulo: la crítica función de enrutamiento de la capa de red. Tanto si la capa de red proporciona un servicio de datagramas (en cuyo caso diferentes paquetes intercambiados entre una pareja determinada origen-destino pueden tomar distintas rutas) como un servicio de circuito virtual (en cuyo caso todos los paquetes intercambiados entre un origen y un destino dados seguirán la misma ruta), la capa de red tiene que determinar la ruta que seguirán los paquetes desde los emisores a los receptores. Veremos que el trabajo del enrutamiento consiste en determinar buenas rutas desde los emisores hasta los receptores a través de la red de routers.

Normalmente, un host está conectado directamente a un router, el **router predeterminado** para el host (también denominado **router del primer salto** para el host). Cuando un host envía un paquete, éste se transfiere a su router predeterminado. Nos referiremos al router predeterminado del host de origen como **router de origen** y al router predeterminado del host de destino como **router de destino**. El problema de enrutar un paquete desde el host de origen al host de destino, evidentemente, se reduce al problema de enrutar el paquete desde el router de origen al router de destino, que es el foco de esta sección.

El propósito de un algoritmo de enrutamiento es por tanto muy simple: dado un conjunto de routers, con enlaces que conectan dichos routers, un algoritmo de enrutamiento determina una "buena" ruta desde el router de origen al router de destino. Normalmente, una buena ruta es aquella que tiene el coste mínimo. Sin embargo, veremos que, en la práctica, los problemas del mundo real, como las políticas utilizadas (por ejemplo, una regla que establezca que "el router *x,* que pertenece a la organización *Y,* no debería reenviar ningún paquete cuyo origen sea la red de la organización *Z*"), también entran en juego para compli-

car los conceptualmente simples y elegantes algoritmos cuya teoría subyace a las prácticas de enrutamiento utilizadas en las redes actuales.

Para formular los problemas de enrutamiento se utilizan grafos. Recuerde que un **grafo** $G = (N,E)$ es un conjunto N de nodos y una colección E de aristas, donde cada arista es una pareja de nodos de N. En el contexto del enrutamiento de la capa de red, los nodos del grafo representan los routers (los puntos en los que se toman las decisiones acerca del reenvío de los paquetes) y las aristas que conectan estos nodos representan los enlaces físicos entre los routers. En la Figura 4.27 se muestra la abstracción de grafo de una red de computadoras. Si desea ver algunos grafos que representan mapas de redes reales, consulte [Dodge 2007, Cheswick 2000]; para ver un estudio de cómo diferentes modelos basados en grafos permiten modelar Internet, consulte [Zegura 1997, Faloutsos 1999, Li 2004].

Como se muestra en la Figura 4.27, una arista también tiene un valor que representa su coste. Normalmente, el coste de una arista puede reflejar la longitud física del enlace correspondiente (por ejemplo, un enlace transoceánico tendría un coste mayor que un enlace terrestre de corto alcance), la velocidad del enlace o el coste monetario asociado con el enlace. Para nuestros propósitos, simplemente utilizaremos el coste del enlace como algo que nos viene dado, sin preocuparnos por cómo se determina. Para cualquier arista (x,y) de E, designamos $c(x,y)$ como el coste de la arista entre los nodos x e y. Si el par (x,y) no pertenece a E, hacemos $c(x,y) = \infty$. Sólo vamos a considerar los grafos no dirigidos (es decir, grafos cuyas aristas no tienen una dirección), de modo que la arista (x,y) es la misma que la arista (y,x) y además $c(x,y) = c(y,x)$. Además, un nodo y se dice que es un **vecino** del nodo x si (x,y) pertenece a E.

Dado que las distintas aristas de la abstracción de grafo tienen costes asignados, un objetivo natural de un algoritmo de enrutamiento es identificar las rutas de coste mínimo entre los orígenes y los destinos. Con el fin de definir este problema de manera más precisa, recuerde que una **ruta** en un grafo $G = (N,E)$ es una secuencia de nodos $(x_1, x_2,..., x_p)$ tal que cada una de las parejas (x_1,x_2), (x_2,x_3),...,(x_{p-1},x_p) son aristas pertenecientes a E. El coste de una ruta $(x_1,x_2,..., x_p)$ es simplemente la suma del coste de todas las aristas a lo largo de la ruta; es decir, $c(x_1,x_2) + c(x_2,x_3) + ...+ c(x_{p-1},x_p)$. Dados dos nodos cualesquiera x e y, normalmente existen muchas rutas entre los dos nodos, teniendo cada una de ellas un coste. Una o más de estas rutas será una **ruta de coste mínimo**. Por tanto, el problema del coste mínimo está claro: hallar una ruta entre el origen y el destino que tenga el coste mínimo. Por ejemplo, en la Figura 4.27 la ruta de coste mínimo entre el nodo de origen u y el nodo de destino

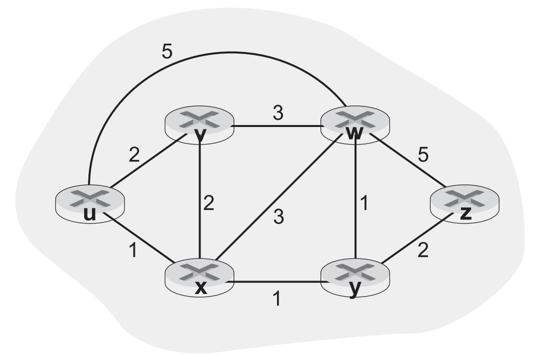


Figura 4.27 • Modelo de grafo de una red de computadoras.

w es (*u, x, y, w*) con un coste de ruta igual a 3. Observe que si todas las aristas del grafo tienen el mismo coste, la ruta de coste mínimo es también la **ruta más corta** (es decir, la ruta con el número mínimo de enlaces entre el origen y el destino).

Como ejercicio, intente encontrar la ruta de coste mínimo desde el nodo *u* al *z* en la Figura 4.27 y reflexione sobre cómo ha calculado esa ruta. Si es usted como la mayor parte de la gente, habrá determinado la ruta de *u* a *z* examinando la Figura 4.27, trazando unas pocas rutas de *u* a *z*, y llegando de alguna manera al convencimiento de que la ruta que ha elegido es la de coste mínimo de entre todas las posibles rutas (¿ha comprobado las 17 rutas posibles entre *u* y *z*? ¡Probablemente no!). Este cálculo es un ejemplo de un algoritmo de enrutamiento centralizado (el algoritmo de enrutamiento se ejecutó en un lugar, su cerebro, que dispone de la información completa de la red). En términos generales, una forma de clasificar los algoritmos de enrutamiento es dependiendo de si son globales o descentralizados.

- Un **algoritmo de enrutamiento global** calcula la ruta de coste mínimo entre un origen y un destino utilizando el conocimiento global y completo acerca de la red. Es decir, el algoritmo toma como entradas la conectividad entre todos los nodos y todos los costes de enlace. Esto requiere por tanto que el algoritmo de alguna forma obtenga esta información antes de realizar realmente el cálculo. El cálculo en sí puede hacerse en un sitio (un algoritmo de enrutamiento global centralizado) o replicarse en varios sitios. La característica distintiva aquí, sin embargo, es que un algoritmo global dispone de toda la información acerca de la conectividad y de los costes de los enlaces. En la práctica, los algoritmos con información de estado global a menudo se denominan **algoritmos de estado de enlaces (LS, *Link-State*)**, ya que el algoritmo tiene que ser consciente del coste de cada enlace de la red. Estudiaremos los algoritmos LS en la Sección 4.5.1.
- En un **algoritmo de enrutamiento descentralizado**, el cálculo de la ruta de coste mínimo se realiza de manera iterativa y distribuida. Ningún nodo tiene toda la información acerca del coste de todos los enlaces de la red. En lugar de ello, al principio, cada nodo sólo conoce los costes de sus propios enlaces directamente conectados. Después, a través de un proceso iterativo de cálculo e intercambio de información con sus nodos vecinos (es decir, los nodos que están en el otro extremo de los enlaces a los que él mismo está conectado), cada nodo calcula gradualmente la ruta de coste mínimo hacia un destino o conjunto de destinos. El algoritmo de enrutamiento descentralizado que estudiaremos más adelante en la Sección 4.5.2 se denomina **algoritmo de vector de distancias** (DV, *Distance-Vector*), porque cada nodo mantiene un vector de estimaciones de los costes (distancias) a todos los demás nodos de la red.

Una segunda forma general de clasificar los algoritmos de enrutamiento es según sean estáticos o dinámicos. En los **algoritmos de enrutamiento estático**, las rutas cambian muy lentamente con el tiempo, con frecuencia como resultado de una intervención humana (por ejemplo, una persona que edita manualmente la tabla de reenvío de un router). Los **algoritmos de enrutamiento dinámico** modifican los caminos de enrutamiento a medida que la carga de tráfico o la topología de la red cambian. Un algoritmo dinámico puede ejecutarse periódicamente o como respuesta directa a cambios en la topología o en el coste de los enlaces. Aunque los algoritmos dinámicos responden mejor a los cambios de la red, también son más susceptibles a problemas como los bucles de enrutamiento y a la oscilación de rutas.

Una tercera forma de clasificar los algoritmos de enrutamiento es según sean sensibles o no a la carga. En un **algoritmo sensible a la carga**, los costes de enlace varían de forma dinámica para reflejar el nivel actual de congestión en el enlace subyacente. Si se asocia un

coste alto con un enlace que actualmente esté congestionado, el algoritmo de enrutamiento tenderá a elegir rutas que eviten tal enlace congestionado. Aunque los primeros algoritmos de enrutamiento de ARPAnet eran sensibles a la carga [McQuillan 1980], se encontró una serie de dificultades [Huitema 1998]. Los algoritmos de enrutamiento actuales de Internet (como RIP, OSPF y BGP) no son sensibles a la carga, ya que el coste de un enlace no refleja explícitamente su nivel actual (o reciente) de congestión.

4.5.1 Algoritmo de enrutamiento de estado de enlaces (LS)

Recuerde que en un algoritmo de estado de enlaces, la topología de la red y el coste de todos los enlaces son conocidos; es decir, están disponibles como entradas para el algoritmo LS. En la práctica, esto se consigue haciendo que cada nodo difunda paquetes del estado de los enlaces a *todos* los demás nodos de la red, con cada paquete de estado de enlace conteniendo las identidades y los costes de sus enlaces conectados. En la práctica (por ejemplo, con el protocolo de enrutamiento OSPF de Internet, que estudiaremos en la Sección 4.6.1), esto suele conseguirse mediante un algoritmo **de difusión de estado de enlaces** [Perlman 1999]. Veremos los algoritmos de difusión en la Sección 4.7. El resultado de difundir la información de los nodos es que todos los nodos tienen una visión completa e idéntica de la red. Cada nodo puede entonces ejecutar el algoritmo LS y calcular el mismo conjunto de rutas de coste mínimo que cualquier otro nodo.

El algoritmo de enrutamiento de estado de enlaces que presentamos a continuación se conoce como *algoritmo de Dijkstra*, en honor a su inventor. Un algorimo estrechamente relacionado es el algoritmo de Prim; consulte [Cormen 2001] para ver una exposición de carácter general sobre los algoritmos de grafos. El algoritmo de Dijkstra calcula la ruta de coste mínimo desde un nodo (el origen, al que denominaremos u) hasta todos los demás nodos de la red. El algoritmo de Dijkstra es iterativo y tiene la propiedad de que después de la *k-ésima* iteración del algoritmo, se conocen las rutas de coste mínimo hacia k nodos de destino y entre las rutas de coste mínimo a todos los nodos de destino, estas k rutas tendrán los k costes más pequeños. Definimos la siguiente notación:

- $D(v)$: coste de la ruta de coste mínimo desde el nodo de origen al destino v, para esta iteración del algoritmo.
- $p(v)$: nodo anterior (vecino de v) a lo largo de la ruta de coste mínimo desde el origen hasta v.
- N': subconjunto de nodos; v pertenece a N' si la ruta de coste mínimo desde el origen hasta v se conoce de forma definitiva.

El algoritmo de enrutamiento global consta de un paso de inicialización seguido de un bucle. El número de veces que se ejecuta el bucle es igual al número de nodos de la red. Al terminar, el algoritmo habrá calculado las rutas más cortas desde el nodo de origen u hasta cualquier otro nodo de la red.

Algoritmo de estado de enlaces (LS) para el nodo de origen *u*

```
Inicialización:
  N’ = {u}
  for todo nodo v
    if v es un vecino de u
```

```
5            then D(v) = c(u,v)
6          else D(v) = ∞
7
8     Bucle
9         encontrar w no perteneciente a N' tal que D(w) sea un mínimo
10        sumar w a N'
11        actualizar D(v) para cada vecino v de w, que no pertenezca a N':
12             D(v) = min( D(v), D(w) + c(w,v) )
13        /* el nuevo coste a v es o bien el antiguo coste a v o el coste
14         de la ruta de coste mínimo a w más el coste desde w a v */
15    until N'= N
```

Por ejemplo, considere la red de la Figura 4.27 y calcule las rutas de coste mínimo desde *u* a todos los destinos posibles. En la Tabla 4.3 se muestra una tabla de resumen de los cálculos del algoritmo, donde cada línea de la tabla proporciona los valores de las variables del algoritmo al final de la iteración. Veamos en detalle los primeros pasos.

- En el paso de inicialización, las rutas de coste mínimo actualmente conocidas desde *u* a sus vecinos conectados directamente, *v, x* y *w,* se inicializan a 2, 1 y 5, respectivamente. En particular, fíjese en que el coste a *w* es igual a 5 (aunque pronto veremos que, de hecho, no existe una ruta de coste más pequeño), ya que éste es el coste del enlace directo (un salto) de *u* a *w.* Los costes a *y* y *z* se hacen igual a infinito porque no están directamente conectados a *u.*
- En la primera iteración, buscamos entre aquellos nodos que todavía no se han añadido al conjunto N' y localizamos el nodo que tiene el coste mínimo después de finalizar la iteración previa. Dicho nodo es *x,* con un coste de 1, y por tanto *x* se añade al conjunto N'. La línea 12 del algoritmo LS se ejecuta entonces para actualizar $D(v)$ para todos los nodos *v,* proporcionando los resultados mostrados en la segunda línea (Paso 1) de la Tabla 4.3. El coste de la ruta a *v* no varía. Se determina que el coste de la ruta a *w* (que era 5 al final de la inicialización) a través del nodo *x* tiene un coste de 4. Así, se selecciona esta ruta de coste mínimo y el predecesor de *w* a lo largo de la ruta más corta desde *u* se establece en *x*. De forma similar, se calcula el coste a *y* (a través de *x*) y se obtiene que es 2, y la tabla se actualiza de acuerdo con ello.
- En la segunda iteración, se determina que los nodos *v* e *y* tienen las rutas de coste mínimo (2), y deshacemos el empate arbitrariamente, añadiendo *y* al conjunto N', de modo que ahora N' contiene a *u*, *x* e *y*. El coste de los restantes nodos que todavía no pertenecen a N', es decir, los nodos *v*, *w* y *z*, se actualiza en la línea 12 del algoritmo LS, dando los resultados mostrados en la tercera fila de la Tabla 4.3.
- Y así sucesivamente. . . .

Cuando el algoritmo LS termina, tenemos para cada nodo su predecesor a lo largo de la ruta de coste mínimo desde el nodo de origen. Para cada predecesor, también tenemos *su* predecesor, y así de este modo podemos construir la ruta completa desde el origen a todos los destinos. La tabla de reenvío de un nodo, por ejemplo, del nodo *u*, puede entonces reconstruirse a partir de esta información almacenando, para cada destino, el nodo del siguiente salto en la ruta de coste mínimo desde *u* al destino. La Figura 4.28 muestra las rutas de coste mínimo resultantes y la tabla de reenvío de *u* para la red de la Figura 4.27.

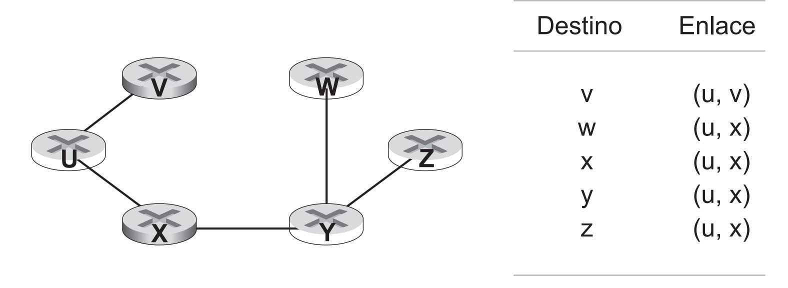
paso	*N′*	*D(v),p(v)*	*D(w),p(w)*	*D(x),p(x)*	*D(y),p(y)*	*D(z),p(z)*
0	u	2,u	5,u	1,u	∞	∞
1	ux	2,u	4,x		2,x	∞
2	uxy	2,u	3,y			4,y
3	uxyv		3,y			4,y
4	uxyvw					4,y
5	uxyvwz					

Tabla 4.3 • Ejecución del algoritmo de estado de enlaces para la red de la Figura 4.27.

¿Cuál es la complejidad de cálculo de este algoritmo? Es decir, dados *n* nodos (sin contar el origen) ¿en el caso peor, cuántos cálculos hay que realizar para determinar las rutas de coste mínimo desde el origen a todos los destinos? En la primera iteración, tenemos que buscar a través de los *n* nodos para determinar el nodo *w* que no pertenece a *N′* y que tiene el coste mínimo. En la segunda iteración, tenemos que comprobar *n* – 1 nodos para determinar el coste mínimo; en la tercera iteración, hay que comprobar *n* – 2 nodos, y así sucesivamente. En general, el número total de nodos a través de los que tenemos que buscar teniendo en cuenta todas las iteraciones es igual a $n(n + 1)/2$ y, por tanto, decimos que la implementación anterior del algoritmo LS tiene, en el caso peor, una complejidad de orden *n* al cuadrado: $O(n^2)$. (Una implementación más sofisticada de este algoritmo, que utiliza una estructura de datos conocida como montón (*heap*), puede calcular el mínimo en la línea 9 en tiempo logarítmico en lugar de lineal, reduciendo así la complejidad.)

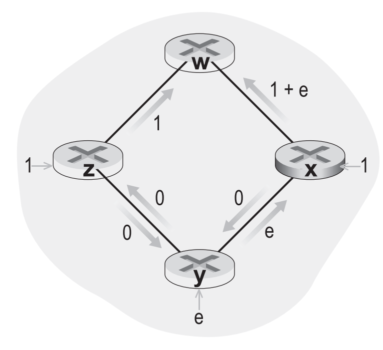
Antes de terminar nuestro estudio del algoritmo LS, consideremos una patología que puede surgir. La Figura 4.29 muestra una topología de red simple donde el coste de cada enlace es igual a la carga transportada por el enlace, reflejando, por ejemplo, el retardo experimentado en ese enlace. En este ejemplo, los costes de los enlaces no son simétricos; es decir, *c*(*u,v*) es igual a *c*(*v,u*) sólo si la carga transportada en ambas direcciones del enlace (*u,v*) es la misma. En este ejemplo, el nodo *z* origina una unidad de tráfico destinada a *w,* el nodo *x* también origina una unidad de tráfico destinada a *w,* y el nodo *y* inyecta una cantidad de tráfico igual a *e,* también destinado a *w*. El enrutamiento inicial se muestra en la Figura 4.29(a) con los costes de enlace correspondientes a la cantidad de tráfico transportado.

Cuando se vuelve a ejecutar el algoritmo LS, el nodo *y* determina (basándose en los costes de enlace mostrados en la Figura 4.29(a)) que la ruta en sentido horario a *w* tiene un

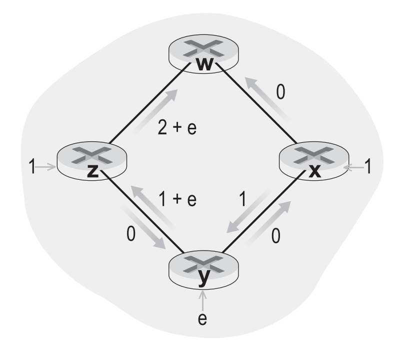


Destino	Enlace
v	(u, v)
w	(u, x)
x	(u, x)
y	(u, x)
z	(u, x)

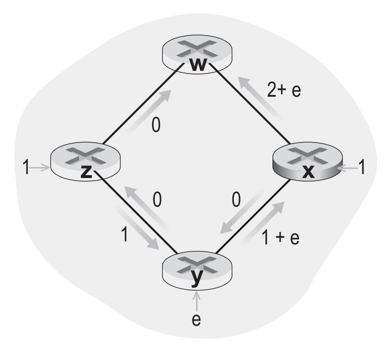
Figura 4.28 • Rutas de coste mínimo y tabla de reenvío para el nodo u.



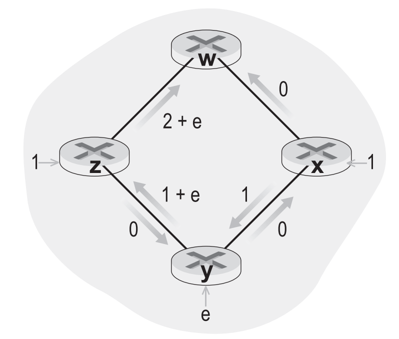
a. Enrutamiento inicial



b. *x*, *y* descubren la mejor ruta hacia *w*, en sentido horario



c. *x*, *y*, *z* detectan la mejor ruta hacia *w*, en sentido antihorario



d. *x*, *y*, *z*, detectan la mejor ruta hacia *w*, en sentido horario

Figura 4.29 • Oscilaciones con enrutamiento sensible a la congestión.

coste de 1, mientras que la ruta en sentido antihorario a *w* (que era la que había estado utilizando) tiene un coste de 1 + *e*. Por tanto, la ruta de coste mínimo de *y* a *w* ahora va en sentido horario. De forma similar, *x* determina que ahora su nueva ruta de coste mínimo a *w* va en sentido horario, dando como resultado los costes mostrados en la Figura 4.29(b). Cuando el algoritmo LS se ejecuta otra vez, los nodos *x, y* y *z* detectan una ruta de coste cero a *w* en el sentido antihorario y todos ellos envían su tráfico a las rutas en sentido antihorario. La siguiente vez que se ejecuta el algoritmo LS, *x, y* y *z* enrutan su tráfico a las rutas en sentido horario.

¿Qué podemos hacer para evitar tales oscilaciones (que pueden producirse en cualquier algoritmo que utilice una métrica de enlace basada en la congestión o el retardo, no sólo en un algoritmo LS)? Una solución sería obligar a que los costes de enlace no dependieran de la cantidad de tráfico transportado (una solución inaceptable, ya que uno de los objetivos del enrutamiento es evitar los enlaces altamente congestionados; por ejemplo, los enlaces con un alto retardo). Otra solución consiste en garantizar que no todos los routers ejecuten el algoritmo LS al mismo tiempo. Ésta parece una solución más razonable, ya que podríamos esperar que, aunque los routers ejecuten el algoritmo LS con la misma periodicidad, la instancia de ejecución del algoritmo no sería la misma en cada uno de los nodos. Es interesante ver que los investigadores han comprobado que los rou-

ters de Internet pueden auto-sincronizarse entre ellos [Floyd Synchronization 1994]. Es decir, incluso aunque inicialmente ejecuten el algoritmo con el mismo periodo pero en distintos instantes de tiempo, la instancia de ejecución del algoritmo puede llegar a sincronizarse en los routers y permanecer sincronizada. Una forma de evitar tal auto-sincronización es que cada router elija aleatoriamente el instante en el que enviar un anuncio de enlace.

Ahora que ya hemos estudiado el algoritmo LS, vamos a abordar otro algoritmo de enrutamiento importante que se utiliza actualmente en la práctica: el algoritmo de enrutamiento por vector de distancias.

4.5.2 Algoritmo de enrutamiento por vector de distancias (DV)

Mientras que el algoritmo LS es un algoritmo que emplea información global, el algoritmo por **vector de distancias** (**DV**) es iterativo, asíncrono y distribuido. Es *distribuido* en el sentido de que cada nodo recibe información de uno o más de sus vecinos *directamente conectados*, realiza un cálculo y luego distribuye los resultados de su cálculo de vuelta a sus vecinos. Es *iterativo* porque este proceso continúa hasta que no hay disponible más informacion para ser intercambiada entre los vecinos. (Además, el algoritmo también finaliza por sí mismo, es decir, no existe ninguna señal que indique que los cálculos deberían detenerse; simplemente se detienen.) El algoritmo es *asíncrono,* en el sentido de que no requiere que todos los nodos operen sincronizados entre sí. Como tendremos oportunidad de ver, un algoritmo asíncrono, iterativo, distribuido y que finaliza por sí mismo es mucho más interesante y divertido que un algoritmo centralizado.

Antes de presentar el algoritmo de vector de distancias, es conveniente que veamos una relación importante que existe entre los costes de las rutas de coste mínimo. Sea $d_x(y)$ el coste de la ruta de coste mínimo desde el nodo x al nodo y. Entonces, los costes mínimos están relacionados mediante la conocida ecuación de Bellman-Ford,

$$d_x(y) = \text{mín}_v\{c(x,v) + d_v(y)\}, \qquad (4.1)$$

donde min_v se calcula para todos los vecinos de x. La ecuación de Bellman-Ford es bastante intuitiva. De hecho, después de viajar de x a v, si tomamos la ruta de coste mínimo de v a y, el coste de la ruta es $c(x,v) + d_v(y)$. Puesto que hay que comenzar viajando a algún vecino v, el coste mínimo de x a y será el mínimo de $c(x,v) + d_v(y)$, calculado para todos los vecinos v.

Pero para aquellos que sean escépticos en cuanto a la validez de la ecuación, vamos a comprobarla para el nodo de origen u y el nodo de destino z de la Figura 4.27. El nodo de origen u tiene tres vecinos: los nodos v, x y w. Recorriendo las distintas rutas del grafo, es fácil ver que $d_v(z) = 5$, $d_x(z) = 3$ y $d_w(z) = 3$. Introduciendo estos valores en la Ecuación 4.1, junto con los costes $c(u,v) = 2$, $c(u,x) = 1$ y $c(u,w) = 5$, se obtiene $d_u(z) = \text{mín}\{2 + 5, 5 + 3, 1 + 3\} = 4$, que obviamente es cierto y es exactamente lo que nos proporciona el algoritmo de Dijskstra para la misma red. Esta rápida verificación debería ayudarle a disipar cualquier duda que pueda tener.

La ecuación de Bellman-Ford no es únicamente una curiosidad intelectual; realmente tiene una importancia práctica significativa. En concreto, la solución de la ecuación de Bellman-Ford proporciona las entradas de la tabla de reenvío del nodo x. Para ver esto, sea v^* cualquier nodo vecino que alcanza el mínimo dado por la Ecuación 4.1. Entonces, si el nodo x desea enviar un paquete al nodo y a lo largo de la ruta de coste mínimo, debería en primer lugar reenviar el paquete al nodo v^*. Por tanto, la tabla de reenvío del nodo x especificaría el

nodo v^* como el router del siguiente salto para el destino final y. Otra importante contribución práctica de la ecuación de Bellman-Ford es que sugiere la forma en que tendrá lugar la comunicación vecino a vecino en el algoritmo de vector de distancias.

La idea básica es la siguiente: cada nodo x comienza con $D_x(y)$, una estimación del coste de la ruta de coste mínimo desde sí mismo al nodo y, para todos los nodos de N. Sea $\mathbf{D}_x = [D_x(y)$: y perteneciente a $N]$ el vector de distancias del nodo x, que es el vector de coste estimado desde x a todos los demás nodos y pertenecientes a N. Con el algoritmo de vector de distancias, cada nodo x mantiene la siguiente información de enrutamiento:

- Para cada vecino v, el coste $c(x,v)$ desde x al vecino directamente conectado, v.
- El vector de distancias del nodo x, es decir, $\mathbf{D}_x = [D_x(y)$: y perteneciente a $N]$, que contiene la estimación que x hace de su coste hacia todos los destinos y de N.
- Los vectores de distancias de cada uno de sus vecinos, es decir, $\mathbf{D}_v = [D_v(y)$: y perteneciente a $N]$ para cada vecino v de x.

En el algoritmo asíncrono distribuido, de vez en cuando, cada nodo envía una copia de su vector de distancias a cada uno de sus vecinos. Cuando un nodo x recibe un nuevo vector de distancias procedente de cualquiera de sus vecinos v, guarda dicho vector de v y luego utiliza la ecuación de Bellman-Ford para actualizar su propio vector de distancias como sigue:

$$D_x(y) \quad \text{mín}_v\{c(x,v) + D_v(y)\} \quad \text{para cada nodo } y \text{ de } N$$

Si el vector de distancias del nodo x ha cambiado como resultado de este paso de actualización, entonces el nodo x enviará su vector de distancias actualizado a cada uno de sus vecinos, lo que puede a su vez actualizar sus propios vectores distancia. De forma bastante milagrosa, siempre y cuando todos los nodos continúen intercambiando sus vectores distancia de forma asíncrona, cada coste estimado $D_x(y)$ converge a $d_x(y)$, el coste real de la ruta de coste mínimo del nodo x al nodo y [Bertsekas 1991].

Algoritmo por vector de distancias (DV)

En cada nodo x:

```
Inicialización:
   for todos los destinos y pertenecientes a N:
      D_x(y) = c(x,y)   /* si y no es un vecino, entonces c(x,y) = ∞ */
   for cada vecino w
      D_w(y) = ? para todos los destinos y pertenecientes a N
   for cada vecino w
      enviar vector de distancias D_x = [D_x(y): y perteneciente N] a w

bucle
   wait (hasta ver una variación en el coste de enlace de un vecino w
         o hasta recibir un vector de distancias de algún vecino w)

   for cada y perteneciente a N:
      D_x(y) = min_v{c(x,v) + D_v(y)}

   if D_x(y) varía para cualquier destino y
```

```
17          enviar vector de distancia Dx = [Dx(y): y perteneciente N] a
18          todos los vecinos
19
20    forever
```

En el algoritmo de vector de distancias, un nodo *x* actualiza la estimación de su vector de distancias si se produce un cambio en el coste de uno de sus enlaces directamente conectados o si recibe una actualización de un vector de distancias de algún vecino. Pero para actualizar su propia tabla de reenvío para un determinado destino *y*, lo que realmente necesita saber el nodo *x* no es la distancia de la ruta más corta a *y*, sino el nodo vecino $v^*(y)$, que es el router del siguiente salto a lo largo de la ruta más corta a *y*. Como es lógico, el router del siguiente salto $v^*(y)$ es el vecino *v* que alcanza el mínimo en la línea 14 del algoritmo DV. (Si existen varios vecinos *v* que alcanzan el mínimo, entonces $v^*(y)$ puede ser cualquiera de esos vecinos.) Por tanto, en la líneas 13–14, para cada destino *y*, el nodo *x* también determina $v^*(y)$ y actualiza su tabla de reenvío para el destino *y*.

Recuerde que al algoritmo LS es un algoritmo global en el sentido de que requiere que cada nodo obtenga en primer lugar un mapa completo de la red antes de ejecutar el algoritmo de Dijkstra. El algoritmo DV es un algoritmo *descentralizado* y no utiliza dicha información global. De hecho, la única información que tendrá un nodo es el coste de los enlaces a los vecinos a los que está directamente conectado y la que recibe de esos vecinos. Cada nodo espera una actualización de cualquier vecino (líneas 10–11), calcula su nuevo vector de distancias cuando recibe una actualización (línea 14) y distribuye su nuevo vector de distancias a sus vecinos (líneas 16–18). En la práctica, los algoritmos del tipo vector de distancias se utilizan en muchos protocolos de enrutamiento, entre los que se incluyen RIP y BGP, ISO IDRP, Novell IPX y el ARPAnet original.

La Figura 4.30 ilustra el funcionamiento del algoritmo DV para el caso de la red simple de tres nodos mostrada en la parte superior de la figura. El funcionamiento del algoritmo se ilustra para el caso síncrono, donde todos los nodos reciben simultáneamente vectores distancia de sus vecinos, calculan sus nuevos vectores distancia e informan a sus vecinos si sus vectores de distancias han cambiado. Después de estudiar este ejemplo, puede comprobar usted mismo que el algoritmo también funciona correctamente en el modo asíncrono, es decir, efectuándose cálculos en los nodos y generándose y recibiéndose actualizaciones en cualquier instante.

La columna más a la izquierda de la figura muestra tres **tablas de enrutamiento** iniciales para cada uno de los tres nodos. Por ejemplo, la tabla de la esquina superior izquierda corresponde a la tabla de enrutamiento inicial del nodo *x*. Dentro de una tabla de enrutamiento concreta, cada fila es un vector de distancias (específicamente, la tabla de enrutamiento de cada nodo incluye su propio vector de distancias y el de cada uno de sus vecinos). Por tanto, la primera fila de la tabla de enrutamiento inicial del nodo *x* es $\boldsymbol{D}_x = [D_x(x), D_x(y), D_x(z)] = [0, 2, 7]$. La segunda y tercera filas de esta tabla son los vectores distancia recibidos más recientemente de los nodos *y* y *z*, respectivamente. Puesto que durante la inicialización el nodo *x* no ha recibido nada aún del nodo *y* ni del *z*, las entradas de la segunda y de la tercera filas se inicializan con infinito.

Después de la inicialización, cada nodo envía su vector de distancias a cada uno de sus dos vecinos. Esto se indica en la Figura 4.30 mediante las flechas que salen de la primera columna de las tablas y van hasta la segunda columna. Por ejemplo, el nodo *x* envía su vector de distancias $\boldsymbol{D}_x = [0, 2, 7]$ a los nodos *y* y *z*. Después de recibir las actualizaciones, cada nodo recalcula su propio vector de distancias. Por ejemplo, el nodo *x* calcula

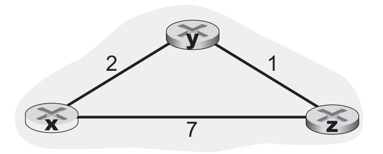


Tabla nodo x

desde	coste a x	y	z
x	0	2	7
y	∞	∞	∞
z	∞	∞	∞

desde	coste a x	y	z
x	0	2	3
y	2	0	1
z	7	1	0

desde	coste a x	y	z
x	0	2	3
y	2	0	1
z	3	1	0

Tabla nodo y

desde	coste a x	y	z
x	∞	∞	∞
y	2	0	1
z	∞	∞	∞

desde	coste a x	y	z
x	0	2	7
y	2	0	1
z	7	1	0

desde	coste a x	y	z
x	0	2	3
y	2	0	1
z	3	1	0

Tabla nodo z

desde	coste a x	y	z
x	∞	∞	∞
y	∞	∞	∞
z	7	1	0

desde	coste a x	y	z
x	0	2	7
y	2	0	1
z	3	1	0

desde	coste a x	y	z
x	0	2	3
y	2	0	1
z	3	1	0

Tiempo

Figura 4.30 • Algoritmo de vector de distancias (DV).

$$D_x(x) = 0$$

$$D_x(y) = \text{mín}\{c(x,y) + D_y(y),\ c(x,z) + D_z(y)\} = \text{mín}\{2 + 0,\ 7 + 1\} = 2$$

$$D_x(z) = \text{mín}\{c(x,y) + D_y(z),\ c(x,z) + D_z(z)\} = \text{mín}\{2 + 1,\ 7 + 0\} = 3$$

Por tanto, la segunda columna muestra, para cada nodo, el nuevo vector de distancias del nodo junto con los vectores distancia que acaba de recibir de sus vecinos. Observe, por ejemplo, que la estimación del nodo *x* para el coste mínimo al nodo *z*, $D_x(z)$, ha cambiado de 7 a 3. Fíjese también en que para el nodo *x,* el nodo vecino *y* alcanza el mínimo en la línea 14 del algoritmo de vector de distancias; luego en esa etapa del algoritmo tenemos que, en el nodo *x*, $v^*(y) = y$ y $v^*(z) = y$.

Una vez que los nodos recalculan sus vectores distancia, envían de nuevo los valores actualizados a sus vecinos (si se ha producido un cambio). Esto se indica en la Figura 4.30

mediante las flechas que van desde la segunda columna a la tercera columna de las tablas. Observe que únicamente los nodos *x* y *z* envían actualizaciones: el vector de distancias del nodo *y* no ha cambiado, por lo que no envía ninguna actualización. Después de recibir las actualizaciones, los nodos recalculan sus vectores distancia y actualizan sus tablas de enrutamiento, lo que se muestra en la tercera columna.

El proceso de recibir vectores distancia actualizados de los vecinos, recalcular las entradas de la tabla de enrutamiento e informar a los vecinos de los costes modificados de la ruta de coste mínimo hacia un destino continúa hasta que ya no se envían mensajes de actualización. En esta situación, puesto que no se envían mensajes de actualización, no se realizarán más cálculos de la tabla de enrutamiento y el algoritmo entrará en un estado de reposo; es decir, todos los nodos se encontrarán a la espera indicada por las líneas 10–11 del algoritmo del vector de distancias. El algoritmo permanece en el estado de reposo hasta que el coste de un enlace cambia, como se explica a continuación.

Algoritmo de vector de distancias: cambios en el coste de los enlaces y fallo de los enlaces

Cuando un nodo que ejecuta el algoritmo DV detecta un cambio en el coste del enlace desde sí mismo a un vecino (líneas 10–11), actualiza su vector de distancias (líneas 13–14) y, si existe un cambio en el coste de la ruta de coste mínimo, informa a sus vecinos (líneas 16–18) de su nuevo vector de distancias. La Figura 4.31(a) ilustra un escenario en el que el coste del enlace de *y* a *x* cambia de 4 a 1. Aquí vamos a centrarnos únicamente en las entradas de la tabla de distancias de *y* y *z* al destino *x*. El algoritmo de vector de distancias da lugar a la siguiente secuencia de sucesos:

- En el instante t_0, *y* detecta el cambio en el coste del enlace (el coste ha cambiado de 4 a 1), actualiza su vector de distancias e informa a sus vecinos de este cambio, puesto que su vector de distancias ha cambiado.
- En el instante t_1, *z* recibe la actualización de *y* y actualiza su tabla. Calcula el nuevo coste mínimo a *x* (ha disminuido de un coste de 5 a un coste de 2) y envía su nuevo vector de distancias a sus vecinos.
- En el instante t_2, *y* recibe la actualización de *z* y actualiza su tabla de distancias. El coste mínimo de *y* no cambia y, por tanto, *y* no envía ningún mensaje a *z*. El algoritmo entra en el estado de reposo.

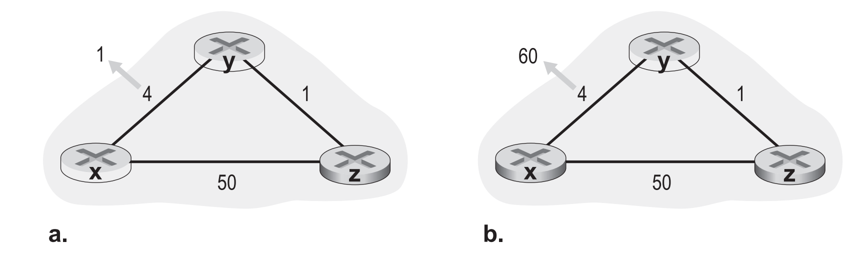


Figura 4.31 • Cambios en el coste del enlace.

Así, sólo se han necesitado dos iteraciones para que el algoritmo DV alcance un estado de reposo. Las buenas noticias acerca de la disminución del coste entre x e y se han propagado rápidamente a través de la red.

Consideremos ahora lo que ocurre cuando el coste de un enlace *aumenta.* Suponga que el coste del enlace entre x e y aumenta de 4 a 60, como se muestra en la Figura 4.31(b).

1. Antes de que el coste del enlace varíe, $D_y(x) = 4$, $D_y(z) = 1$, $D_z(y) = 1$ y $D_z(x) = 5$. En el instante t_0, y detecta el cambio en el coste del enlace (el coste ha variado de 4 a 60). El nodo y calcula su nueva ruta de coste mínimo a x, obteniendo un coste de:

$$D_y(x) = \text{mín}\{c(y,x) + D_x(x),\ c(y,z) + D_z(x)\} = \text{mín}\{60 + 0,\ 1 + 5\} = 6$$

   Por supuesto, con nuestra visión global de la red podemos ver que este nuevo coste a través de z es *erróneo.* Pero la única información que tiene el nodo y es que su coste directo a x es 60 y que z le ha dicho a y que z podría alcanzar x con un coste de 5. Por tanto, para llegar a x, y ahora enrutaría a través de z, confiando plenamente en que z será capaz de llegar hasta x con un coste de 5. En t_1 tenemos por tanto un **bucle de enrutamiento** (para llegar a x, y enruta a través de z, y z enruta a través de y). Un bucle de enrutamiento es como un agujero negro (un paquete destinado a x que llega a y o z en t_1 rebotará entre estos dos nodos permanentemente, o hasta que las tablas de reenvío cambien).
2. Dado que el nodo y ha calculado un nuevo coste mínimo a x, informa a z de este nuevo vector de distancias en el instante t_1.
3. En algún momento posterior a t_1, z recibe un nuevo vector de distancias de y, que indica que el coste mínimo de y a x es 6. z sabe que puede llegar a y con un coste de 1 y, por tanto, calcula un nuevo coste mínimo a x de $D_z(x) = \min\{50 + 0, 1 + 6\} = 7$. Puesto que el coste mínimo de z a x ha aumentado, entonces informa a y de su nuevo vector de distancias en t_2.
4. De forma similar, después de recibir el nuevo vector de distancias de z, y determina que $D_y(x) = 8$ y envía a z su vector de distancias. El nodo z determina entonces que $D_z(x) = 9$ y envía a y su vector de distancias, y así sucesivamente.

¿Durante cuánto tiempo continuará el proceso? Puede comprobar por sí mismo que el bucle se mantendrá durante 44 iteraciones (intercambios de mensajes entre y y z), hasta que z finalmente calcule que el coste de su ruta a través de y es mayor que 50. En esta situación, z (¡finalmente!) determinará que su ruta de coste mínimo a x es a través de su conexión directa con x. El nodo y entonces enrutará hacia x a través de z. Como puede ver, las malas noticias acerca del aumento del coste del enlace han tardado mucho en propagarse. ¿Qué habría ocurrido si el coste del enlace $c(y, x)$ hubiera cambiado de 4 a 10.000 y el coste $c(z, x)$ hubiera sido 9.999? A causa de los escenarios de este tipo, en ocasiones se designa a este problema con el nombre de problema de la cuenta hasta infinito.

Algoritmo de vector de distancias: técnica de la inversa envenenada

El escenario concreto de los bucles que acabamos de describir puede evitarse utilizando una técnica conocida como inversa envenenada (*poisoned reverse*). La idea es simple: si z enruta a través de y para llegar al destino x, entonces z anunciará a y que su distancia a x es infinita, es decir, z anunciará a y que $D_z(x) = \infty$ (incluso aunque z sepa que, en realidad, $D_z(x) = 5$). z mantendrá esta pequeña mentira destinada a y mientras continúe enrutando

hacia x a través de y. Dado que y cree que z no dispone de una ruta hacia x, el nodo y nunca intentará enrutar hacia x a través de z, siempre que z continúe enrutando hacia x a través de y (y mintiendo acerca de ello).

Veamos ahora cómo la inversa envenenada resuelve el problema concreto del bucle encontrado antes en la Figura 4.31(b). Como resultado de la inversa envenenada, la tabla de distancias de y indica que $D_z(x) = \infty$. Cuando el coste del enlace (x, y) cambia de 4 a 60 en el instante t_0, y actualiza su tabla y continúa enrutando directamente a x, a pesar del muy alto coste de 60, e informa a z de su nuevo coste a x, es decir, $D_y(x) = 60$. Después de recibir la actualización en t_1, z cambia inmediatamente su ruta a x para que sea a través del enlace directo (z, x) con un coste de 50. Puesto que ésta es la nueva ruta de coste mínimo a x, y dado que la ruta ya no pasa a través de y, ahora en t_2, z informa a y de que $D_z(x) = 50$. Después de recibir la actualización de z, y actualiza su tabla de distancias con $D_y(x) = 51$. Además, dado que ahora z está en la ruta de coste mínimo de y a x, el nodo y envenena la ruta inversa de z a x, informando a z en el instante t_3 de que $D_y(x) = \infty$ (aunque y sepa que, en realidad, $D_y(x) = 51$).

¿Resuelve la inversa envenenada el problema general de la cuenta hasta infinito? No. Puede comprobar que los bucles que implican a tres o más nodos (en lugar de simplemente a dos nodos vecinos) no serán detectados por la técnica de la inversa envenenada.

Comparación de los algoritmos de enrutamiento LS y DV

Los algoritmos de vector de distancias y de estado de enlaces utilizan métodos complementarios para el cálculo de las rutas. Con el algoritmo de vector de distancias, cada nodo *sólo* se comunica con sus vecinos directamente conectados, y les proporciona sus estimaciones de coste mínimo desde sí mismo a *todos* los demás nodos (conocidos) de la red. En el algoritmo de estado de enlaces, cada nodo se comunica con *todos* los restantes nodos (vía difusión), pero *sólo* les informa de los costes de sus enlaces directamente conectados. Vamos a terminar este estudio sobre los algoritmos de estado de enlaces y de vector de distancias haciendo una rápida comparación de algunos de sus atributos. Recuerde que N es el conjunto de nodos (routers) y E es el conjunto de aristas (enlaces).

- *Complejidad del mensaje.* Hemos visto que el algoritmo LS requiere que cada nodo conozca el coste de cada enlace de la red. Esto requiere el envío de $O(|N|\,|E|)$ mensajes. Además, cuando el coste de un enlace cambia, el nuevo coste tiene que enviarse a todos los nodos. El algoritmo de vector de distancias requiere intercambios de mensajes entre los vecinos directamente conectados en cada iteración. Hemos visto que el tiempo necesario para que el algoritmo converja puede depender de muchos factores. Cuando los costes de los enlaces cambian, el algoritmo de vector de distancias propagará los resultados del coste del enlace que ha cambiado sólo si el nuevo coste de enlace da lugar a una ruta de coste mínimo distinta para uno de los nodos conectados a dicho enlace.
- *Velocidad de convergencia.* Hemos visto que nuestra implementación del algoritmo de estado de enlaces es un algoritmo $O(|N|^2)$ que requiere enviar $O(|N|\,|E|)$) mensajes. El algoritmo de vector de distancias puede converger lentamente y pueden aparecer bucles de enrutamiento mientras está convergiendo. Este algoritmo también sufre el problema de la cuenta hasta infinito.
- *Robustez.* ¿Qué puede ocurrir si un router falla, funciona mal o es saboteado? Con el algoritmo de estado de enlaces, un router podría difundir un coste incorrecto para uno de sus enlaces conectados (pero no para los otros). Un nodo también podría corromper o eli-

minar cualquier paquete recibido como parte de un mensaje de difusión LS. Pero, con el algoritmo LS, un nodo sólo calcula su propia tabla de reenvío, mientras que otros nodos realizan cálculos similares por sí mismos. Esto significa que los cálculos de rutas son algo independientes en LS, proporcionando un mayor grado de robustez. Con el algoritmo de vector de distancias, un nodo puede anunciar rutas de coste mínimo incorrectas a cualquiera o a todos los destinos. (De hecho, en 1997, un router que funcionaba mal en un pequeño ISP proporcionó a los routers troncales nacionales información de enrutamiento errónea. Esto hizo que otros routers inundaran con una gran cantidad de tráfico al router que funcionaba mal e hizo que amplias partes de Internet estuvieran desconectadas durante varias horas [Neumann 1997].) En un sentido más general, observamos que, en cada iteración, los cálculos de un nodo con el algoritmo de vector de distancias se pasan a sus vecinos y luego, indirectamente, al vecino del vecino en la siguiente iteración. En este sentido, con el algoritmo de vector de distancias, un cálculo de nodo incorrecto puede difundirse a través de toda la red.

En resumen, ningún algoritmo es el ganador evidente; de hecho, ambos algoritmos se utilizan en Internet.

Otros algoritmos de enrutamiento

Los algoritmos LS y DV que hemos estudiado no sólo se emplean ampliamente en la práctica, sino que además son prácticamente los *únicos* algoritmos de enrutamiento empleados actualmente en Internet. No obstante, los investigadores han propuesto muchos algoritmos de enrutamiento a lo largo de los últimos 30 años, desde algunos extremadamente simples hasta otros realmente complejos y sofisticados. Una clase bastante amplia de algoritmos de enrutamiento está basada en interpretar el tráfico de paquetes como una serie de flujos entre los orígenes y los destinos de una red. Con este método, el problema del enrutamiento puede formularse matemáticamente como un problema de optimización con restricciones, conocido como problema de flujo en una red [Bertsekas 1991]. Hay otro conjunto más de algoritmos de enrutamiento que son los que se derivan del mundo de la telefonía. Estos **algoritmos basados en la conmutación de circuitos** son interesantes para las redes de datos de conmutación de paquetes en aquellos casos en los que hay que reservar recursos en cada enlace (por ejemplo, buffers, o una fracción del ancho de banda del enlace) para cada conexión enrutada a través del enlace. Aunque la formulación del problema del enrutamiento podría parecer bastante diferente de la formulación del enrutamiento de coste mínimo que hemos visto en este capítulo, existen un gran número de similitudes, al menos en lo que respecta al algoritmo de determinación de la ruta (algoritmo de enrutamiento). Consulte [Ash 1998; Ross 1995; Girard 1990] para obtener información detallada acerca de esta área de investigación.

4.5.3 Enrutamiento jerárquico

En nuestro estudio de los algoritmos de estado de enlaces y de vector de distancias, hemos visto la red simplemente como una colección de routers interconectados. Un router era indistinguible de otro en el sentido de que los routers ejecutaban el mismo algoritmo de enrutamiento para calcular las rutas a través de la red completa. En la práctica, este modelo y la imagen de un conjunto homogéneo de routers que ejecutan todos ellos el mismo algoritmo de enrutamiento es un poco simplista por al menos dos razones importantes:

- *Escala.* Cuando el número de routers comienza a hacerse grande, la sobrecarga implicada en los cálculos, el almacenamiento y la comunicación de la información de enrutamiento (por ejemplo, las actualizaciones o los cambios en las rutas de coste mínimo de LS) se hace prohibitiva. Actualmente, Internet consta de cientos de millones de hosts. Almacenar la información de enrutamiento de cada uno de estos hosts evidentemente requeriría enormes cantidades de memoria. La sobrecarga requerida para difundir las actualizaciones LS entre todos los routers de Internet no dejaría ancho de banda disponible para la transmisión de paquetes de datos. Seguramente, un algoritmo de vector de distancias que iterara entre tal enorme cantidad de routers nunca llegaría a converger. Evidentemente, es preciso hacer algo para reducir la complejidad del cálculo de rutas en redes tan grandes como Internet.
- *Autonomía administrativa.* Aunque los investigadores tienden a ignorar problemas tales como el deseo de las empresas de operar sus routers a su antojo (por ejemplo, emplear cualquier algoritmo de enrutamiento que elijan) o de ocultar ciertos aspectos de la organización interna de su red al mundo exterior, éstas son consideraciones extremadamente importantes. Idealmente, una organización debería poder operar y administrar su red como deseara, siempre que sea posible conectar su red a otras redes externas.

Estos dos problemas pueden resolverse organizando los routers en **sistemas autónomos (AS, *Autonomous System*)**, con cada AS formado por un grupo de routers que normalmente se encuentran bajo el mismo control administrativo (por ejemplo, operados por el mismo ISP o pertenecientes a la misma red empresarial). Los routers de un mismo AS ejecutan todos ellos el mismo algoritmo de enrutamiento (por ejemplo, un algoritmo LS o DV) y disponen de información acerca de ellos (exactamente como en el caso de nuestro modelo ideal de la sección anterior). El algoritmo de enrutamiento que se ejecuta dentro de un sistema autónomo se conoce como **protocolo de enrutamiento interno del sistema autónomo**. Por supuesto, será necesario conectar los sistemas autónomos entre sí y, luego, uno o más de los routers de un sistema autónomo tendrá la tarea adicional de ser responsables del reenvío de paquetes a los destinos externos al sistema autónomo; estos routers se conocen como **routers gateway** (o de pasarela).

La Figura 4.32 proporciona un ejemplo simple con tres sistemas autónomos: AS1, AS2 y AS3. En esta figura, las líneas más gruesas representan las conexiones de los enlaces directos entre parejas de routers. Las líneas más finas que salen de los routers representan subredes que están conectadas directamente a los routers. AS1 tiene cuatro routers (1a, 1b, 1c y 1d) que ejecutan el protocolo de enrutamiento interno del sistema autónomo utilizado dentro de AS1. Por tanto, cada uno de estos cuatro routers sabe cómo reenviar paquetes a lo largo de la ruta óptima a cualquier destino dentro de AS1. De forma similar, los sistemas autónomos AS2 y AS3 tienen tres routers. Observe que el protocolo de enrutamiento interno del sistema autónomo en los sistemas AS1, AS2 y AS3 no tiene por qué ser el mismo. Observe también que los routers 1b, 1c, 2a y 3a son todos ellos routers pasarela.

Ahora debería estar claro cómo los routers de un sistema autónomo determinan las rutas para las parejas origen-destino que son internas al AS. Pero todavía nos falta una pieza importante en el enrutamiento terminal a terminal. ¿Cómo sabe un router de un AS enrutar un paquete cuyo destino se encuentra fuera del sistema autónomo? Responder a esta pregunta es fácil si el AS sólo dispone de un router de pasarela para conectarse a otro sistema autónomo. En este caso, puesto que el algoritmo de enrutamiento interno del sistema autónomo ha determinado la ruta de coste mínimo desde cada router interno al router de pasa-

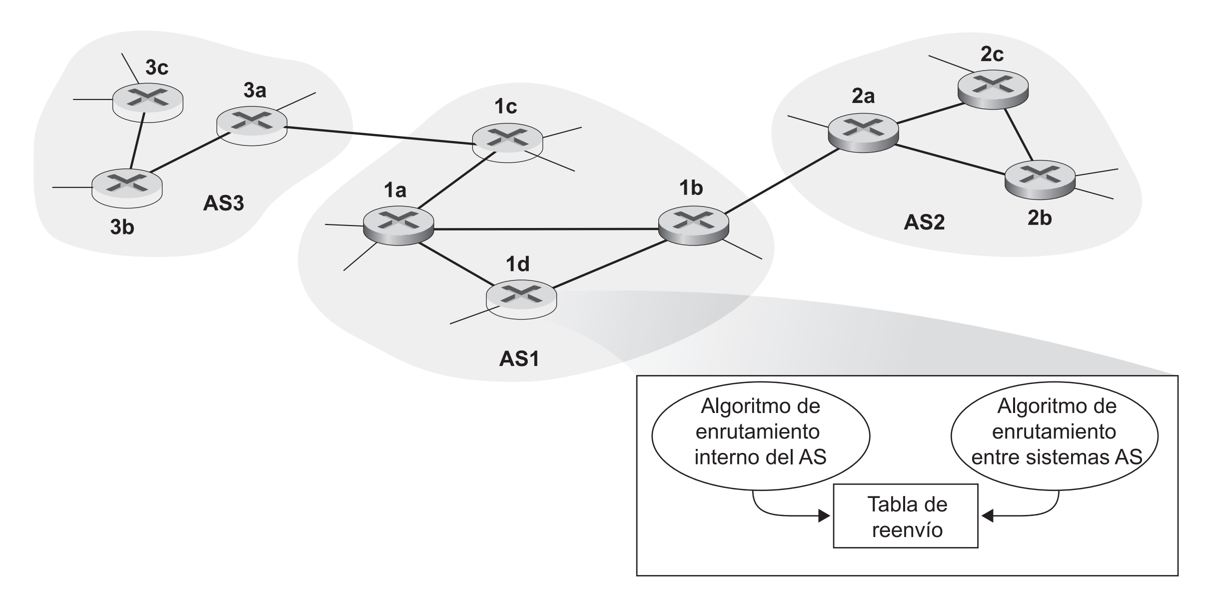


Figura 4.32 • Ejemplo de sistemas autónomos interconectados.

rela, cada router interno sabe cómo debe reenviar el paquete. El router de pasarela, una vez que ha recibido el paquete, lo reenvía al enlace que conecta con el exterior del AS. El sistema autónomo al otro lado del enlace toma entonces la responsabilidad de enrutar el paquete a su destino final. Por ejemplo, suponga que el router 2b de la Figura 4.32 recibe un paquete cuyo destino está fuera de AS2. El router 2b reenviará el paquete bien al router 2a o al 2c, tal y como se especifica en la tabla de reenvío del router 2b, que ha sido configurada por el protocolo de enrutamiento interno de AS2. El paquete finalmente llegará al router de pasarela 2a, el cual reenviará el paquete a 1b. Una vez que el paquete ha abandonado el router 2a, el trabajo que AS2 tenía que hacer con ese paquete está hecho.

Por tanto, el problema es fácil cuando el sistema autónomo de origen sólo tiene un enlace que le comunica con el exterior. Pero, ¿qué ocurre si el AS de origen tiene dos o más enlaces (a través de dos o más routers de pasarela) que le llevan fuera del AS? Entonces el problema de saber dónde reenviar el paquete es significativamente más complicado. Por ejemplo, considere un router de AS1 y suponga que recibe un paquete cuyo destino está fuera del sistema autónomo. Evidentemente, el router tiene que reenviar el paquete a uno de sus dos routers de pasarela, 1b o 1c, pero ¿a cuál? Para resolver este problema, AS1 tiene que (1) aprender qué destinos son alcanzables a través de AS2 y qué destinos son alcanzables a través de AS3 y (2) propagar esa información de alcanzabilidad a todos los routers de AS1, de manera que cada router pueda configurar su tabla de reenvío para gestionar los destinos externos al AS. Estas dos tareas (obtener la información de alcanzabilidad de los AS vecinos y propagar dicha información de alcanzabilidad a todos los routers internos del AS) son realizadas por el **protocolo de enrutamiento entre sistemas autónomos**. Puesto que el protocolo de enrutamiento interno del sistema autónomo implica la comunicación entre dos AS, los dos AS que van a comunicarse tienen que utilizar el mismo protocolo de enrutamiento interno. De hecho, en Internet, todos los sistemas autónomos ejecutan el mismo protocolo de enrutamiento interno, el protocolo BGP4, que veremos en la siguiente sección. Como se muestra en la Figura 4.32, cada router recibe información de un protocolo de enrutamiento

interno de sistema autónomo y de un protocolo de enrutamiento entre sistemas autónomos, y emplea la información de ambos protocolos para configurar su tabla de reenvío.

Veamos un ejemplo. Considere una subred *x* (identificada por su dirección CIDR), y suponga que AS1 aprende del protocolo de enrutamiento interno del AS que la subred *x* es alcanzable desde AS3 pero *no* desde AS2. AS1 propaga entonces esta información a todos sus routers. Cuando el router 1d aprende que la subred *x* es alcanzable desde AS3 y, por tanto, desde el router de pasarela 1c, determina, a partir de la información proporcionada por el protocolo de enrutamiento interno del AS, la interfaz del router que está en la ruta de coste mínimo desde el router 1d hasta el router de pasarela 1c. Supongamos que se trata de la interfaz *I*. El router 1d puede entonces incluir la entrada (*x, I*) en su tabla de reenvío. (Este ejemplo, y otros presentados en esta sección, proporciona sólo ideas generales, pero es una simplificación de lo que realmente ocurre en Internet. En la siguiente sección ofrecemos una descripción más detallada, aunque más compleja, al abordar el protocolo BGP.)

Siguiendo con el ejemplo anterior, supongamos ahora que AS2 y AS3 están conectados a otros sistemas autónomos, los cuales no se han mostrado en el diagrama. Suponga también que AS1 aprende del protocolo de enrutamiento interno que la subred *x* es alcanzable tanto desde AS2, a través del router de pasarela 1b, como desde AS3, a través del router de pasarela 1c. AS1 tendría entonces que propagar esta información a todos sus routers, incluyendo al router 1d. Para configurar su tabla de reenvío, el router 1d tendría que determinar a qué router de pasarela, 1b o 1c, enviará los paquetes cuyo destino sea la subred *x*. Un método, que es el que se suele utilizar en la práctica, consistiría en utilizar el **enrutamiento de la patata caliente**. Con este tipo de enrutamiento, el sistema autónomo suelta el paquete (la patata caliente) tan rápido como sea posible (de forma más precisa, de la forma más barata posible). Para hacer esto, el router envía el paquete al router de pasarela que tiene el coste más pequeño (desde el router de origen hasta el router de pasarela) de entre todos los routers de pasarela que cuentan con una ruta hasta ese destino. En nuestro ejemplo, el enrutamiento de la patata caliente, que se ejecuta en 1d, utilizaría la información del protocolo de enrutamiento interno del AS para determinar los costes de las rutas a 1b y 1c, y luego seleccionaría la ruta con el coste mínimo. Una vez que se ha elegido la ruta, el router 1d añade una entrada para la subred *x* en su tabla de reenvío. La Figura 4.33 resume las acciones realizadas en el router 1d para añadir una nueva entrada para *x* en la tabla de reenvío.

Cuando un sistema autónomo obtiene información acerca de un destino de un AS vecino, puede anunciar esta información de enrutamiento a algunos de sus otros sistemas autónomos vecinos. Por ejemplo, suponga que AS1 aprende de AS2 que la subred *x* es alcanzable a través de AS2. AS1 podría entonces comunicar a AS3 que *x* es alcanzable a través de AS1. De esta forma, si AS3 necesita enrutar un paquete destinado a *x*, AS3 reenviará el paquete a AS1, el cual a su vez lo reenviará a AS2. Como veremos al estudiar el protocolo

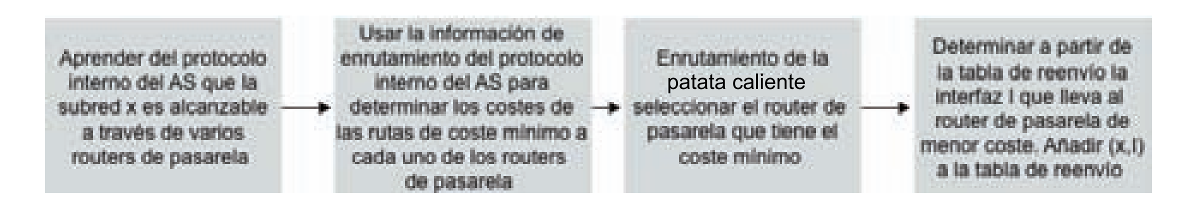


Figura 4.33 • Pasos para añadir a la tabla de reenvío de un router un destino externo al sistema autónomo.

BGP, un sistema autónomo tiene bastante flexibilidad a la hora de decidir qué destinos anunciar a sus sistemas autónomos vecinos. Se trata de una decisión *política*, que normalmente depende más de cuestiones económicas que de cuestiones técnicas.

Recuerde de la Sección 1.5 que Internet está formado por una jerarquía de proveedores ISP interconectados. Luego, ¿cuál es la relación entre los ISP y los sistemas autónomos? Podría pensarse que los routers de un ISP y los enlaces que los interconectan constituyen un sistema autónomo. Aunque éste suele ser el caso, muchos ISP dividen su red en varios sistemas autónomos. Por ejemplo, algunos ISP de nivel 1 utilizan un único sistema autónomo para toda su red y otros dividen su red en decenas de sistemas autónomos interconectados.

Resumiento, los problemas de escala y autoridad administrativa se resuelven definiendo sistemas autónomos. Dentro de un sistema autónomo, todos los routers ejecutan el mismo protocolo de enrutamiento interno. Entre ellos, los AS ejecutan el mismo protocolo de enrutamiento entre AS. El problema de la escala se resuelve de manera que un router interno de un AS sólo necesita tener información acerca de los routers de su AS. El problema de la autoridad administrativa está resuelto, ya que una organización puede ejecutar el protocolo de enrutamiento interno de sistema autónomo que prefiera; sin embargo, cada pareja de sistemas autónomos conectados deben ejecutar el mismo protocolo de enrutamiento entre sistemas autónomos, con el fin de poder intercambiar información de alcanzabilidad.

En la siguiente sección examinaremos dos protocolos de enrutamiento internos de sistema autónomo (RIP y OSPF) y el protocolo de enrutamiento entre sistemas autónomos (BGP) que se emplean actualmente en Internet. Estos casos de estudio nos permitirán completar adecuadamente nuestro análisis del enrutamiento jerárquico.

4.6 Enrutamiento en Internet

Una vez estudiados el direccionamiento en Internet y el protocolo IP, vamos a pasar a los protocolos de enrutamiento de Internet; su trabajo consiste en determinar la ruta que sigue un datagrama entre un origen y un destino. Veremos que los protocolos de enrutamiento de Internet se basan en muchos de los principios que hemos aprendido anteriormente en el capítulo. Los algoritmos de estado de enlaces y de vector de distancias estudiados en las Secciones 4.5.1 y 4.5.2, y el concepto de sistema autónomo tratado en la Sección 4.5.3 son fundamentales para la forma en que se lleva a cabo el enrutamiento en la red Internet actual.

Recuerde de la Sección 4.5.3 que un sistema autónomo (AS) es una colección de routers bajo el mismo control técnico y administrativo, y que todos ellos ejecutan el mismo protocolo de enrutamiento para comunicarse entre sí. Cada sistema autónomo, a su vez, normalmente contiene varias subredes (entendiendo el término subred en el sentido, muy preciso, de direccionamiento, tal como se explica en la Sección 4.4.2).

4.6.1 Enrutamiento interno de un sistema autónomo de Internet: RIP

Un protocolo de enrutamiento interno de un AS se utiliza para determinar cómo se lleva a cabo el enrutamiento dentro de un sistema autónomo. Los protocolos de enrutamiento internos de los AS se conocen también como **protocolos de pasarela interior**. Históricamente, el enrutamiento dentro de los sistemas autónomos de Internet se ha llevado a cabo principalmente con dos protocolos de enrutamiento: el **Protocolo de información de enrutamiento**

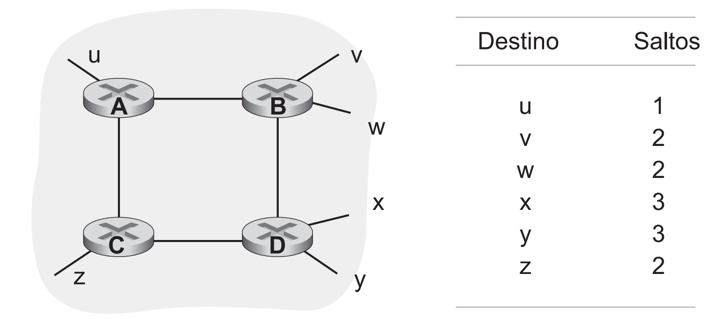
(RIP, *Routing Information Protocol*) y el protocolo **Primero la ruta abierta más corta (OSPF, *Open Shortest Path First*)**. Un protocolo de enrutamiento estrechamente relacionado con OSPF es el protocolo **IS-IS** [RFC 1142, Perlman 1999]. En primer lugar, vamos a ver el protocolo RIP y luego el OSPF.

RIP fue uno de los primeros protocolos de enrutamiento de Internet internos para los AS y todavía hoy es ampliamente utilizado. Sus orígenes están en la arquitectura XNS (*Xerox Network Systems*, Sistemas de red Xerox) a la que debe su nombre. La extensa implantación de RIP se ha debido en gran parte a su inclusión en 1982 en la versión BSD (*Berkeley Software Distribution*) de UNIX que soportaba TCP/IP. La versión 1 de RIP está definida en [RFC 1058], y la versión 2 compatible hacia abajo está definida en [RFC 2453].

RIP es un protocolo de vector de distancias que opera de una forma muy parecida al protocolo DV ideal que hemos examinado en la Sección 4.5.2. La versión de RIP especificada en el documento RFC 1058 utiliza como métrica de coste el recuento de saltos; es decir, cada enlace tiene un coste de 1. En el algoritmo de vector de distancias de la Sección 4.5.2, por simplificar, los costes se definieron entre parejas de routers. En RIP (y también en OSPF), los costes se definen realmente desde el router de origen a una subred de destino. RIP utiliza el término *salto* (*hop*), que es el número de subredes que se atraviesan al seguir la ruta más corta desde el router de origen hasta la subred de destino, incluyendo esta última. La Figura 4.34 ilustra un sistema autónomo con seis subredes terminales. La tabla incluida en la figura indica el número de saltos desde el origen A a cada una de las subredes terminales.

El coste máximo de una ruta está limitado a 15, luego el uso de RIP en los sistemas autónomos está limitado a sistemas autónomos con un diámetro de menos de 15 saltos. Recuerde que en los protocolos de vector de distancias, los routers vecinos intercambian entre sí los vectores distancia. El vector de distancias para cualquier router es la estimación actual de la ruta más corta desde dicho router a las subredes del AS. En RIP, las actualizaciones de enrutamiento son intercambiadas entre los vecinos aproximadamente cada 30 segundos mediante un **mensaje de respuesta RIP**. El mensaje de respuesta enviado por un router o un host contiene una lista de hasta 25 subredes de destino pertenecientes al sistema autónomo, así como la distancia desde el emisor a cada una de esas subredes. Los mensajes de respuesta se conocen como **anuncios RIP**.

Veamos un ejemplo simple de cómo funcionan los anuncios RIP. Considere la parte de un sistema autónomo mostrada en la Figura 4.35. En esta figura, las líneas que conectan los routers indican subredes. Sólo se han etiquetado los routers (*A, B, C* y *D*) y las subredes (*w, x, y* y *z*). Las líneas de puntos indican que el sistema autónomo continúa; por tanto, este sistema autónomo tiene muchos más routers y enlaces que los mostrados.



Destino	Saltos
u	1
v	2
w	2
x	3
y	3
z	2

Figura 4.34 • Número de saltos desde el router de origen A a varias subredes.

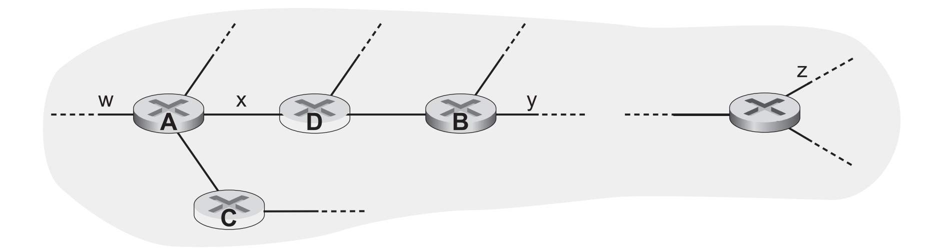


Figura 4.35 • Una parte de un sistema autónomo.

Cada router mantiene una tabla RIP conocida como **tabla de enrutamiento**. La tabla de enrutamiento de un router incluye tanto el vector de distancias del router como la tabla de reenvío del mismo. La Figura 4.36 muestra la tabla de enrutamiento del router *D*. Observe que esta tabla tiene tres columnas. La primera de ellas especifica la subred de destino, la segunda columna detalla la identidad del siguiente router a lo largo de la ruta más corta a la subred de destino y la tercera columna especifica el número de saltos (es decir, el número de subredes que hay que atravesar, incluyendo la subred de destino) para llegar hasta la subred de destino siguiendo el camino más corto. En este ejemplo, la tabla indica que para enviar un datagrama desde el router *D* a la subred de destino *w*, en primer lugar el datagrama tiene primero que reenviarse al router vecino *A;* la tabla también especifica que la subred de destino *w* está a dos saltos por la ruta más corta. Del mismo modo, la tabla indica que la subred *z* está a siete saltos, pasando por el router *B*. En principio, una tabla de enrutamiento tendrá una fila para cada subred que forma parte del sistema autónomo, aunque la versión 2 de RIP permite añadir entradas de subred utilizando técnicas de agregación de rutas similares a las que hemos examinado en la Sección 4.4. La tabla de la Figura 4.36 y las tablas que siguen se han completado parcialmente.

Suponga ahora que 30 segundos más tarde, el router *D* recibe del router *A* el anuncio mostrado en la Figura 4.37. Observe que este anuncio no es otra cosa que la información de la tabla de enrutamiento del router *A*. Esta información indica, en particular, que la subred *z* está sólo a cuatro saltos del router *A*. El router *D,* una vez que ha recibido este anuncio, combina el anuncio (Figura 4.37) con la tabla de enrutamiento antigua (Figura 4.36). En particular, *D* ahora sabe que existe una ruta a través del router *A* a la subred *z* que es más corta que la ruta a través del router *B*. Por tanto, *D* actualiza su tabla de enrutamiento para tener en cuenta la nueva ruta más corta, como se muestra en la Figura 4.38. Se estará preguntando,

Subred de destino	Siguiente router	Número de saltos hasta el destino
w	A	2
y	B	2
z	B	7
x	—	1
. . . .	. . . .	. . . .

Figura 4.36 • Tabla de enrutamiento del router *D* antes de recibir un anuncio del router *A*.

Subred de destino	Siguiente router	Número de saltos hasta el destino
z	C	4
w	—	1
x	—	1
....	....	....

Figura 4.37 • Anuncio del router A.

Subred de destino	Siguiente router	Número de saltos hasta el destino
w	A	2
y	B	2
z	A	5
....	....	....

Figura 4.38 • Tabla de enrutamiento del router *D* después de recibir un anuncio del router *A*.

¿cómo es que la ruta más corta a la subred *z* se ha hecho aún más pequeña? Posiblemente, el algoritmo descentralizado de vector de distancias se encuentra todavía en el proceso de convergencia (véase la Sección 4.5.2), o quizá se han añadido enlaces nuevos y/o routers nuevos al sistema autónomo, cambiando por tanto las rutas más cortas dentro del AS.

Veamos a continuación algunos de los aspectos de implementación de RIP. Recuerde que los routers RIP intercambian anuncios aproximadamente cada 30 segundos. Si un router no tiene noticias de su vecino al menos una vez cada 180 segundos, considera que ese vecino ya no es alcanzable; es decir, o bien ese vecino ha muerto o el enlace que le conectaba con él ha fallado. Si esto ocurre, RIP modifica la tabla de enrutamiento local y luego propaga esta información enviando anuncios a sus routers vecinos (a aquellos que todavía son alcanzables). Un router también puede solicitar información a sus vecinos, mediante un mensaje de solicitud RIP, acerca del coste a un destino dado. Los routers se envían entre sí solicitudes y mensajes de respuesta RIP utilizando UDP en el número de puerto 520. El segmento UDP es transportado entre los routers en un datagrama IP estándar. El hecho de que RIP utilice un protocolo de la capa de transporte (UDP) por encima de un protocolo de la capa de red (IP) para implementar la funcionalidad de la capa de red (un algoritmo de enrutamiento) puede parecer bastante enrevesado (¡y lo es!). Examinar con algo más de profundidad cómo se implementa RIP ayudará a clarificar esta cuestión.

La Figura 4.39 esboza cómo se implementa RIP normalmente en un sistema UNIX; por ejemplo, en una estación de trabajo UNIX que actúa como router. Un proceso conocido como *routed* ejecuta RIP, es decir, mantiene la información de enrutamiento e intercambia mensajes con los procesos *routed* que se ejecutan en los routers vecinos. Dado que RIP se implementa como un proceso de la capa de aplicación (aunque es un proceso muy especial, capaz de manipular las tablas de enrutamiento dentro del kernel de UNIX), puede enviar y recibir mensajes a través de un socket estándar y utilizar un protocolo de transporte estándar. Como se muestra en la figura, RIP se implementa como un protocolo de la capa de aplicación (véase el Capítulo 2) que se ejecuta sobre UDP.

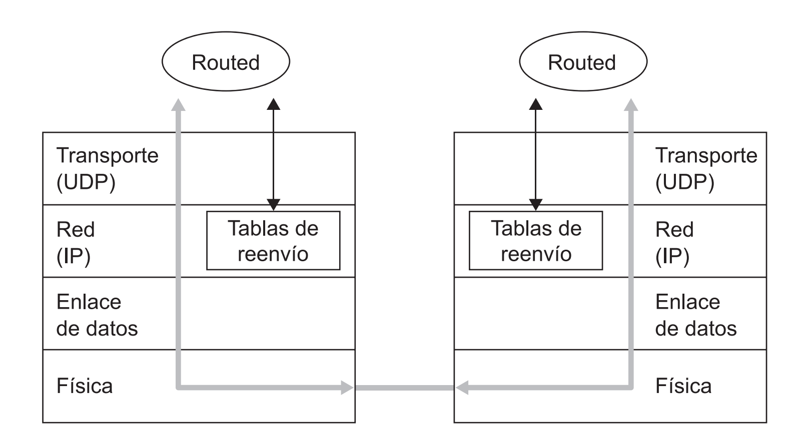


Figura 4.39 • Implementación de RIP mediante el demonio *routed*.

4.6.2 Enrutamiento interno de un AS en Internet: OSPF

Al igual que RIP, el enrutamiento OSPF se utiliza ampliamente para el enrutamiento interno de los sistemas autónomos de Internet. OSPF y su pariente próximo, IS-IS, normalmente se implantan en los ISP de nivel superior, mientras que RIP se implanta en los ISP de nivel inferior y en las redes empresariales. El término "Abierto" (*Open*) de OSPF indica que la especificación del protocolo de enrutamiento está disponible públicamente (por ejemplo, en oposición al protocolo EIGRP de Cisco). La versión más reciente de OSPF, la versión 2, está definida en RFC 2328, un documento público.

OSPF fue concebido como el sucesor de RIP y como tal ofrece una serie de funcionalidades avanzadas. Sin embargo, OSPF es básicamente un protocolo de estado de enlaces que utiliza la técnica de inundación de información de estado de los enlaces y el algoritmo de cálculo de la ruta de coste mínimo de Dijkstra. Con OSPF, un router construye un mapa topológico completo (es decir, un grafo) del sistema autónomo entero. A continuación, el router ejecuta localmente el algoritmo de la ruta más corta de Dijkstra para determinar un árbol de rutas más cortas a todas las *subredes*, con él mismo como nodo raíz. El administrador de la red configura los costes de los enlaces individuales (consulte el recuadro Práctica: configuración de los pesos de los enlaces en OSPF). El administrador puede decidir hacer igual a 1 el coste de todos los enlaces, proporcionando un enrutamiento de número mínimo de saltos, o puede definir los pesos de los enlaces para que sean inversamente proporcionales a la capacidad de los mismos, con el fin de disuadir al tráfico de utilizar los enlaces con pequeño ancho de banda. OSPF no establece una política para definir el peso de los enlaces (esta tarea le corresponde al administrador de la red), sino que proporciona mecanismos (el protocolo) para determinar el enrutamiento de coste mínimo para el conjunto dado de pesos de los enlaces.

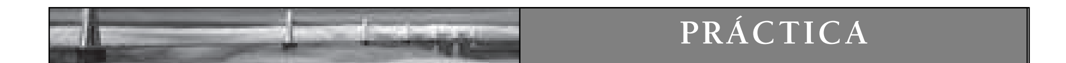
Con OSPF, un router difunde la información de enrutamiento a *todos* los demás routers del sistema autónomo, no sólo a sus routers vecinos. Un router difunde la información de estado de los enlaces cuando se ha producido un cambio en el estado de un enlace (por ejemplo, un cambio en el coste o en su estado activo/inactivo, *up/down*). También difunde periódicamente el estado de un enlace (al menos una vez cada 30 minutos), incluso aunque el estado del mismo no haya cambiado. El documento RFC 2328 destaca que "esta actualización periódica de los anuncios del estado de los enlaces añade robustez al algoritmo LS."

Los anuncios OSPF están contenidos en mensajes OSPF que son transportados directamente por IP, siendo el número del protocolo de la capa superior para OSPF igual a 89. Así, el protocolo OSPF tiene que implementar por sí mismo funcionalidades tales como la de transferencia fiable de mensajes y la de envío de mensajes de difusión acerca del estado de los enlaces. El protocolo OSPF también comprueba que los enlaces estén operativos (mediante un mensaje HELLO que se envía a un vecino conectado) y permite al router OSPF obtener de un vecino la base de datos de estado de los enlaces de toda la red.

Algunas de las funcionalidades avanzadas incluidas en OSPF son las siguientes:

- *Seguridad.* Los intercambios entre routers OSPF (por ejemplo, actualizaciones de estado de los enlaces) pueden ser autenticados. Con la autenticación, sólo pueden participar en el protocolo OSPF los routers de confianza del sistema autónomo, impidiendo así que intrusos maliciosos (o estudiantes de redes que apliquen sus conocimientos recién adquiridos sin permiso) inyecten información incorrecta en las tablas de un router. Por defecto, los paquetes OSPF entre routers no son autenticados y podrían ser alterados. Pueden configurarse dos tipos de mecanismo de autenticación: simple y MD5 (consulte el Capítulo 8 para obtener información sobre MD5 y la autenticación en general). Con la autenticación simple se configura la misma contraseña en todos los routers. Cuando un router envía un paquete OSPF, incluye la contraseña en texto legible. Evidentemente, la autenticación simple no es muy segura. La autenticación MD5 está basada en claves secretas compartidas que están configuradas en todos los routers. Para cada paquete OSPF que se envía, el router calcula el hash MD5 del contenido del paquete OSPF, al que se añade la clave secreta (consulte el Capítulo 7 para ver una explicación acerca de los códigos de autenticación de mensajes). A continuación, el router incluye el valor hash resultante en el paquete OSPF. El router receptor, utilizando la clave secreta preconfigurada, calculará un hash MD5 del paquete y lo comparará con el valor hash que transporta el paquete, verificando de este modo la autenticidad del mismo. En la autenticación MD5 también se utilizan los números de secuencia para protegerse frente a ataques por repetición.
- *Varias rutas de igual coste.* Cuando varias rutas a un destino tienen el mismo coste, OSPF permite utilizar varias rutas (es decir, no es necesario elegir una misma ruta para transportar todo el tráfico, cuando existen varias rutas con igual coste).
- *Soporte integrado para enrutamiento por unidifusión y por multidifusión.* OSPF multidifusión (MOSPF, *Multicast* OSPF) [RFC 1584] añade extensiones simples a OSPF para proporcionar enrutamiento por multidifusión (un tema que cubriremos en detalle en la Sección 4.7.2). MOSPF utiliza la base de datos de enlaces OSPF existente y añade un nuevo tipo de anuncio de estado de enlaces al mecanismo de difusión existente en OSPF para difundir el estado de los enlaces.
- *Soporte para definir una jerarquía dentro de un mismo dominio de enrutamiento.* Quizá el avance más significativo de OSPF sea la capacidad de estructurar los sistemas autónomos de forma jerárquica. En la Sección 4.5.3 ya hemos visto algunas de las ventajas de las estructuras de enrutamiento jerárquicas. Veremos la implementación del enrutamiento jerárquico OSPF a lo largo del resto de esta sección.

Un sistema autónomo OSPF puede configurarse jerárquicamente en áreas. Cada área ejecuta su propio algoritmo de enrutamiento de estado de enlaces OSPF, con cada router de un área difundiendo su estado de enlaces a todos los demás routers de ese área. Dentro de cada área, uno o más **routers de frontera de área** son responsables de enrutar los paquetes



CONFIGURACIÓN DE LOS PESOS DE LOS ENLACES EN OSPF

En nuestra exposición sobre el enrutamiento de estado de enlaces hemos supuesto implícitamente que los pesos de los enlaces están definidos, que se está ejecutando un algoritmo de enrutamiento como OSPF y que el tráfico fluye de acuerdo con las tablas de enrutamiento calculadas por el algoritmo LS. En términos de causa y efecto, los pesos de los enlaces nos vienen dados (es decir, se conocen de antemano) y dan como resultado (mediante el algoritmo de Dijkstra) las rutas que minimizan el coste global. Desde este punto de vista, los pesos de los enlaces reflejan el coste de utilizar un enlace (por ejemplo, si los pesos son inversamente proporcionales a la capacidad, entonces los enlaces de alta capacidad tendrían asociados pesos más pequeños y, por tanto, serían más atractivos desde el punto de vista del enrutamiento) y el algoritmo de Disjkstra sirve para minimizar el coste global.

En la práctica, la relación causa-efecto entre el peso de los enlaces y las rutas puede invertirse, cuando los operadores de red configuran los pesos de los enlaces de manera que se obtengan rutas que permitan alcanzar determinados objetivos de ingeniería de tráfico [Fortz 2000, Fortz 2002]. Por ejemplo, suponga que un operador de red tiene una estimación del flujo de tráfico que entra en la red por cada punto de entrada y que sale de la misma por cada punto de salida. El operador puede entonces implementar un enrutamiento específico para los flujos de entrada-a-salida que minimice la tasa máxima de utilización de los enlaces de la red. Pero con un algoritmo de enrutamiento como OSPF, la principal herramienta de la que dispone el operador para optimizar el enrutamiento de los flujos a través de la red son los pesos de los enlaces. Por tanto, para alcanzar el objetivo de minimizar la tasa máxima de utilización de los enlaces, el operador tiene que encontrar el conjunto de pesos de los enlaces que permita alcanzar este objetivo. Esto constituye una inversión de la relación causa-efecto: el enrutamiento deseado de los flujos es conocido y tienen que determinarse los pesos de los enlaces OSPF, de manera que el algoritmo de enrutamiento OSPF dé como resultado el enrutamiento de flujos deseado.

---

fuera del área. Por último, una única área OSPF del sistema autónomo se configura para actuar como área **troncal (*bakbone*)**. La función principal del área troncal es enrutar el tráfico entre las demás áreas del sistema autónomo. El área troncal siempre contiene a todos los routers de frontera del sistema autónomo y también puede contener routers que no sean de frontera. El enrutamiento entre áreas dentro del sistema autónomo requiere que el paquete sea enrutado en primer lugar a un router de frontera del área (enrutamiento dentro del área) y luego a través del área troncal al router de frontera del área en que se encuentra el destino, para enrutarse por último hacia el destino final.

OSPF es un protocolo relativamente complejo, por lo que aquí lo hemos cubierto brevemente; en [Huitema 1998; Moy 1998; RFC 2328] se proporcionan detalles adicionales.

4.6.3 Enrutamiento entre sistemas autónomos: BGP

Acabamos de ver cómo los ISP utilizan los protocolos RIP y OSPF para determinar las rutas óptimas para las parejas origen-destino que se encuentran en un mismo sistema autónomo.

Ahora vamos a examinar cómo se determinan las rutas para parejas origen-destino que se encuentran en diferentes sistemas autónomos. La versión 4 del **Protocolo de pasarela de frontera (BPG, *Border Gateway Protocol*)**, especificado en el documento RFC 4271 (véase también [RFC 4274; RFC 4276]), es actualmente el protocolo de enrutamiento entre sistemas autónomos estándar *de facto* en Internet. Comúnmente se le conoce como BGP4 o simplemente como **BGP**. Como protocolo de enrutamiento entre sistemas autónomos (véase la Sección 4.5.3), BGP proporciona a cada sistema autónomo mecanismos para:

1. Obtener información acerca de la alcanzabilidad de las subredes de los sistemas autónomos vecinos.
2. Propagar la información de alcanzabilidad a todos los routers internos del sistema autónomo.
3. Determinar "buenas" rutas a las subredes, basándose en la información de alcanzabilidad y en la política del sistema autónomo.

Lo más importante es que BGP permite a cada subred anunciar su existencia al resto de Internet. Una subred vocifera "Existo y estoy aquí", y BGP garantiza que todos los sistemas autónomos de Internet sabrán que la subred existe y cómo llegar a ella. Si no fuera por BGP, las subredes estarían aisladas, resultando desconocidas para el resto de Internet.

Fundamentos de BGP

BGP es extremadamente complejo; se han dedicado libros completos al tema y todavía hay muchas cuestiones que no se comprenden bien [Yannuzzi 2005]. Además, incluso después de haber leído los libros y documentos RFC, es posible que sea difícil dominar completamente BGP si no se ha practicado con él durante muchos meses (por no decir años) como diseñador o administrador de un ISP de nivel superior. No obstante, dado que BGP es un protocolo absolutamente crítico en Internet (básicamente, es el protocolo que permite unir todos los componentes), necesitamos adquirir al menos unos conocimientos rudimentarios acerca de cómo funciona. Comenzaremos describiendo cómo podría funcionar BGP en el contexto de la red de ejemplo de la Figura 4.32 que hemos estudiado anteriormente. En esta descripción, nos basaremos en nuestra exposición acerca del enrutamiento jerárquico de la Sección 4.5.3; le recomendamos que repase dicho material.

En BGP, las parejas de routers intercambian información de enrutamiento a través de conexiones TCP semipermanentes utilizando el puerto 179. Las conexiones TCP semipermanentes para la red de la Figura 4.32 se muestran en la Figura 4.40. Normalmente, existe una conexión TCP BGP para cada enlace que conecta directamente dos routers que se encuentran en dos sistemas autónomos diferentes; así, en la Figura 4.40, existe una conexión TCP entre los routers pasarela 3a y 1c, y otra conexión TCP entre los routers pasarela 1b y 2a. Existen también conexiones TCP BGP semipermanentes entre los routers internos de un sistema autónomo. En particular, la Figura 4.40 muestra una configuración habitual de una conexión TCP para cada pareja de routers internos de un sistema autónomo, creando una malla de conexiones TCP dentro de cada sistema autónomo. Para cada conexión TCP, los dos routers situados en los extremos de la conexión se denominan **pares BGP** y la conexión TCP junto con todos los mensajes BGP enviados a través de la conexión se denomina **sesión BGP**. Además, una sesión BGP que abarca dos sistemas autónomos se conoce como **sesión externa BGP** (**eBGP**) y una sesión BGP entre routers de un mismo sistema autónomo se conoce como **sesión interna BGP** (**iBGP**). En la Figura 4.40, las sesiones eBGP están

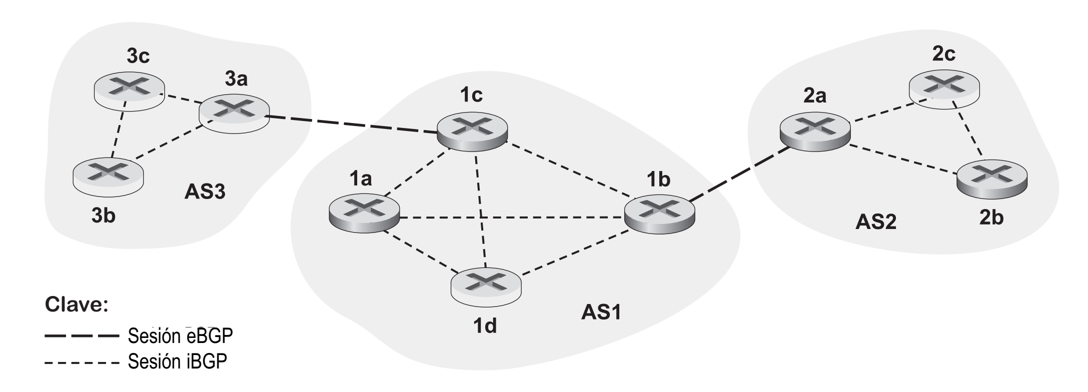


Figura 4.40 • Sesiones eBGP e iBGP.

indicadas mediante líneas de trazo largo y las sesiones iBGP mediante líneas de trazo más corto. Observe que las líneas de sesión BGP de la Figura 4.40 no siempre corresponden a los enlaces físicos de la Figura 4.32.

BGP permite que cada sistema autónomo aprenda qué destinos son alcanzables a través de sus sistemas autónomos vecinos. En BGP, los destinos no son hosts sino **prefijos** CIDR, representando cada prefijo una subred o una colección de subredes. Así, por ejemplo, suponga que hay cuatro subredes conectadas a AS2: 138.16.64/24, 138.16.65/24, 138.16.66/24 y 138.16.67/24. Entonces AS2 podría agrupar los prefijos para estas cuatro subredes y utilizar BGP para anunciar un único prefijo 138.16.64/22 a AS1. Veamos otro ejemplo; suponga que sólo las tres primeras de esas cuatro subredes están en AS2 y que la cuarta subred, 138.16.67/24, está en AS3. Luego, como se describe en el recuadro Práctica de la Sección 4.4.2, dado que los routers utilizan las coincidencias con el prefijo más largo para reenviar los datagramas, AS3 podría anunciar a AS1 el prefijo más específico 138.16.67/24 y AS2 *todavía* podría anunciar a AS1 el prefijo agregado 138.16.64/22.

Examinemos ahora cómo BGP distribuiría la información de alcanzabilidad de prefijos a través de las sesiones BGP mostradas en la Figura 4.40. Como es lógico, utilizando la sesión eBGP entre los routers pasarela 3a y 1c, AS3 envía a AS1 la lista de prefijos que son alcanzables desde AS3; y AS1 envía a AS3 la lista de prefijos que son alcanzables desde AS1. De forma similar, AS1 y AS2 intercambian la información de alcanzabilidad a través de sus routers pasarela 1b y 2a. También, como es de esperar, cuando un router de pasarela (en cualquier sistema autónomo) recibe prefijos aprendidos mediante eBGP, el router de pasarela utiliza sus sesiones iBGP para distribuir los prefijos a los demás routers del sistema autónomo. Por tanto, todos los routers de AS1 aprenden los prefijos de AS3, incluyendo al router de pasarela 1b. Este router (de AS1) puede por tanto volver a anunciar los prefijos de AS3 a AS2. Cuando un router (de pasarela o no) aprende un nuevo prefijo, crea una entrada para el mismo en su tabla de reenvío, como se ha descrito en la Sección 4.5.3.

Atributos de ruta y rutas BGP

Ahora que ya tenemos unos conocimientos básicos sobre BGP, vamos a profundizar un poco en él (omitiendo algunos de los detalles menos importantes). En BGP, un sistema autónomo se identifica mediante su **número de sistema autónomo (ASN, *Autonomous System Number*)** globalmente único [RFC 1930]. (Técnicamente, no todos los sistemas autónomos tie-

nen un ASN. En concreto, un sistema autónomo del tipo terminal (*stub* AS), aquél que sólo transporta el tráfico para el que es origen o destino, normalmente no tendrá un ASN; ignoraremos este tecnicismo en nuestra exposición con el fin de ser más claros.) Los números de AS, al igual que las direcciones IP, son asignados por los registros regionales de la ICANN [ICANN 2009].

Cuando un router anuncia un prefijo en una sesión BGP, incluye con el prefijo una serie de **atributos BGP**. En la jerga de BGP, un prefijo junto con sus atributos se denomina **ruta**. Por tanto, los pares BGP se anuncian rutas entre sí. Dos de los atributos más importantes son AS-PATH y NEXT-HOP:

- *AS-PATH*. Este atributo contiene los sistemas autónomos a través de los que ha pasado el anuncio del prefijo. Cuando se ha pasado un prefijo dentro de un sistema autónomo, el sistema añade su ASN al atributo AS-PATH. Por ejemplo, considere la Figura 4.40 y suponga que el prefijo 138.16.64/24 se anuncia primero desde AS2 a AS1; si a continuación AS1 anuncia el prefijo a AS3, el valor de AS-PATH sería AS2 AS1. Los routers utilizan el atributo AS-PATH para detectar e impedir los bucles de anuncio; en concreto, si un router ve que su sistema autónomo está en la lista de rutas, rechazará el anuncio. Como pronto explicaremos, los routers también utilizan el atributo AS-PATH para seleccionar entre varias rutas hacia el mismo prefijo.
- El atributo NEXT-HOP, que proporciona el enlace crítico entre el protocolo de enrutamiento interno del sistema autónomo y el protocolo de enrutamiento entre sistemas autónomos, tiene un sutil aunque importante uso. *El siguiente salto (NEXT-HOP) es la interfaz de router que inicia la secuencia de sistemas autónomos (AS-PATH)*. Para comprender el uso de este atributo, vamos a hacer referencia de nuevo a la Figura 4.40. Considere lo que ocurre cuando el router de pasarela 3a de AS3 anuncia una ruta al router pasarela 1c de AS1, utilizando una sesión eBGP. La ruta incluye el prefijo anunciado, que denominaremos x, y la secuencia de sistemas autónomos (AS-PATH) que hay que seguir para alcanzar ese prefijo. Este anuncio también incluye el siguiente salto (NEXT-HOP), que es la dirección IP de la interfaz del router 3a que lleva a 1c (recuerde que un router tiene múltiples direcciones IP, una para cada una de sus interfaces.) Consideremos ahora lo que ocurre cuando el router 1d aprende acerca de esta ruta gracias a iBGP. Después de aprender esta ruta a x, el router 1d puede querer reenviar paquetes a x a lo largo de la ruta, es decir, el router 1d puede querer incluir la entrada (x, l) en su tabla de reenvío, donde l es su interfaz que inicia la ruta de coste mínimo desde 1d al router de pasarela 1c. Para determinar l, 1d proporciona la dirección IP en el atributo NEXT-HOP a su módulo de enrutamiento interno del sistema autónomo. Observe que el algoritmo de enrutamiento interno del sistema autónomo ha determinado la ruta de coste mínimo a todas las subredes conectadas a los routers de AS1, incluyendo la subred para el enlace entre 1c y 3a. A partir de esta ruta de coste mínimo desde 1d a la subred 1c-3a, 1d determina su interfaz de router l que da comienzo a esta ruta y luego añade la entrada (x, l) a su tabla de reenvío. En resumen, los routers utilizan el atributo AS-PATH para configurar apropiadamente sus tablas de reenvío.
- La Figura 4.41 ilustra otra situación en la que se necesita el atributo AS-PATH. En esta figura, AS1 y AS2 están conectados mediante dos enlaces entre pares. Un router de AS1 podría aprender dos rutas diferentes hacia el mismo prefijo x. Estas dos rutas podrían seguir la misma secuencia de sistemas autónomos (AS-PATH) hasta x, pero podrían tener distintos valores de NEXT-HOP, correspondientes a los diferentes enlaces entre pares. Uti-

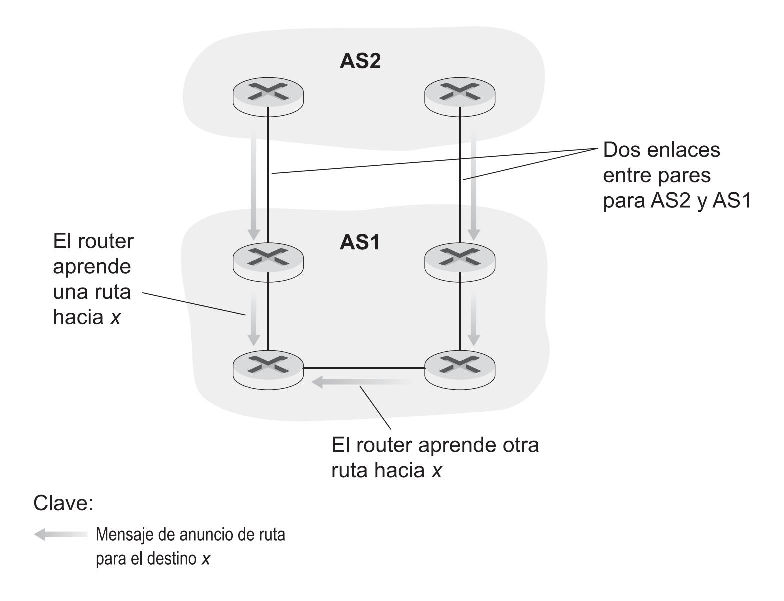


Figura 4.41 • Los atributos NEXT-HOP en los anuncios se utilizan para determinar qué enlace entre pares utilizar.

lizando los valores de AS-PATH y el algoritmo de enrutamiento interno del sistema autónomo, el router puede determinar el coste de la ruta a cada enlace entre pares y luego aplicar el enrutamiento de la patata caliente (véase la Sección 4.5.3) para determinar la interfaz apropiada.

BGP también incluye atributos que permiten a los routers asignar métricas de preferencia a las rutas y un atributo que indica cómo se insertó el prefijo en BGP en el sistema autónomo de origen. Para obtener más información acerca de los atributos de ruta, consulte [Griffin 2009; Stewart 1999; Halabi 2000; Feamster 2004; RFC 4271].

Cuando un router de pasarela recibe un anuncio de un router, utiliza su **política de importación** para decidir si aceptar o filtrar la ruta y si debe definir determinados atributos, como por ejemplo las métricas de preferencia del router. La política de importación puede filtrar una ruta porque el sistema autónomo puede no querer enviar tráfico a través de uno de los sistemas autónomos contenidos en la secuencia AS-PATH de la ruta. El router de pasarela también puede filtrar una ruta porque ya disponga de una ruta preferible hacia el mismo prefijo.

Selección de la ruta BGP

Como se ha descrito anteriormente en esta sección, BGP utiliza las sesiones eBGP e iBGP para distribuir las rutas a todos los routers que forman los sistemas autónomos. A partir de esta distribución, un router puede aprender acerca de más de una ruta a cualquier prefijo, en cuyo caso tendrá que seleccionar una de las posibles rutas. Las entradas para este proceso de selección de ruta es el conjunto de todas las rutas que han sido aprendidas y aceptadas por el router. Si existen dos o más rutas al mismo prefijo, entonces BGP invoca secuencialmente las siguientes reglas de eliminación hasta quedarse con una ruta:

- Se asigna un valor de preferencia local a las rutas, como uno de sus atributos. La preferencia local de una ruta podría haber sido definida por el router o podría haber sido aprendida por otro router perteneciente al mismo sistema autónomo. Ésta es una decisión política que se deja al administrador de red del sistema autónomo (en breve veremos en detalle las políticas de BGP). Se eligen las rutas con los valores de preferencia local más altos.
- De las rutas que quedan (todas ellas con el mismo valor de preferencia local), se selecciona la ruta con el camino de sistemas autónomos (AS-PATH) más corto. Si esta regla fuera la única para seleccionar la ruta, entonces BGP estaría aplicando un algoritmo de vector de distancias para determinar la ruta, siendo la métrica de distancia utilizada el número de saltos entre sistemas autónomos, en lugar del número de saltos entre routers.
- De las restantes rutas (todas con el mismo valor de preferencia local y la misma longitud de AS-PATH), se selecciona la ruta con el router del siguiente salto (NEXT-HOP) más próximo. En este caso, más próximo quiere decir el router para el que el coste de la ruta de coste mínimo, determinado por el algoritmo interno del sistema autónomo, sea más pequeño. Como se ha visto en la Sección 4.5.3, este proceso se conoce como enrutamiento de la patata caliente.
- Si todavía queda más de una ruta, el router utiliza los identificadores BGP para seleccionar la ruta; consulte [Stewart 1999].

Las reglas de eliminación son aun más complejas de lo que acabamos de describir. Para evitar tener pesadillas con BGP, lo mejor es estudiar las reglas de selección de BGP en pequeñas dosis.

Política de enrutamiento

Vamos a ilustrar algunos de los conceptos básicos de la política de enrutamiento de BGP con un ejemplo sencillo. La Figura 4.42 muestra seis sistemas autónomos interconectados: A, B, C, W, X e Y. Es importante observar que A, B, C, W, X e Y son sistemas autónomos, no routers. Supongamos que los sistemas autónomos W, X e Y son redes terminales (*stub*) y que A, B y C son redes troncales de los proveedores. Supongamos también que A, B y C están interconectados entre sí y que proporcionan información completa sobre BGP a sus redes cliente. Todo el tráfico que entra en una **red terminal** tiene que estar destinado a esa red, y todo el tráfico que sale de una red terminal tiene que haber sido originado en esa red. W e Y son claramente redes terminales. X es una **red terminal multiconectada (*multihomed*)**, ya que está conectada al resto de la red a través de dos proveedores diferentes (un escenario cada

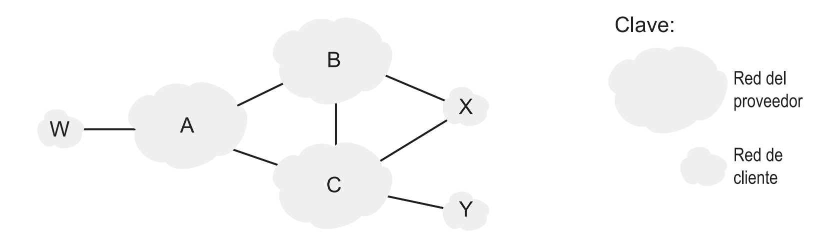
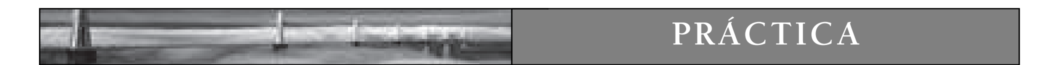


Figura 4.42 • Un escenario BGP simple.



PRÁCTICA

¿POR QUÉ LOS SISTEMAS AUTÓNOMOS DISPONEN DE DIFERENTES PROTOCOLOS DE ENRUTAMIENTO PARA USO INTERNO Y PARA COMUNICARSE ENTRE ELLOS?

Una vez estudiados los detalles de los protocolos de enrutamiento internos y para comunicarse entre sí utilizados por los sistemas autónomos implantados actualmente en Internet, vamos a concluir considerando quizá la cuestión más importante que podríamos plantearnos acerca de estos protocolos (esperamos que le haya surgido esta duda a lo largo del capítulo): ¿por qué se utilizan protocolos de enrutamiento distintos dentro de los sistemas autónomos y para comunicarse entre ellos?

La respuesta a esta pregunta se encuentra en las diferencias entre los objetivos de enrutamiento dentro de un sistema autónomo y entre sistemas autónomos:

- *Política.* Entre sistemas autónomos, prevalecen las políticas. Puede ser importante que el tráfico originado en un determinado sistema autónomo no pueda atravesar otro sistema autónomo específico. De forma similar, un sistema autónomo dado puede desear controlar el tráfico de tránsito transportado entre otros sistemas autónomos que pasa a su través. Hemos visto que BGP transporta atributos de ruta y permite la distribución controlada de información de enrutamiento, de manera que pueden tomarse ese tipo decisiones de enrutamiento basadas en políticas. Dentro de un sistema autónomo, todo está bajo el mismo control administrativo y, por tanto, las políticas desempeñan un papel mucho menos importante en la elección de rutas dentro del sistema autónomo.
- *Escala.* La capacidad de un algoritmo de enrutamiento y de sus estructuras de datos para ampliarse, con el fin de gestionar el enrutamiento hacia/entre una gran cantidad de redes es un problema crítico en el enrutamiento entre sistemas autónomos. Dentro de un sistema autónomo, la escalabilidad es un problema menor, aunque sólo sea porque si un dominio administrativo se hace demasiado grande, siempre es posible dividirlo en dos sistemas autónomos y realizar el enrutamiento entre los dos nuevos sistemas. (Recuerde que OSPF permite crear una jerarquía dividiendo un sistema autónomo en áreas.)
- *Rendimiento.* Puesto que el enrutamiento entre sistemas autónomos está muy orientado a las políticas, la calidad (por ejemplo, el rendimiento) de las rutas utilizadas suele ser una cuestión secundaria (es decir, una ruta más larga o más costosa que satisfaga determinados criterios políticos puede muy bien emplearse antes que una ruta más corta, pero que no cumpla dichos criterios). De hecho, hemos visto que entre sistemas autónomos no existe ni siquiera el concepto de coste asociado con las rutas (salvo por el recuento de saltos entre sistemas autónomos). Sin embargo, dentro de un sistema autónomo, tales cuestiones políticas tienen una importancia menor, lo que permite al enrutamiento centrarse más en el nivel de rendimiento que se puede alcanzar en una ruta.

vez más común en la práctica). Sin embargo, al igual que W e Y, X tiene la caracterísica de que todo el tráfico que entra/sale de X tiene su origen/destino en el propio X. Pero, ¿cómo se puede implementar el comportamiento de esta red terminal? ¿Cómo se impedirá que X reenvíe tráfico entre B y C? Esto se puede conseguir fácilmente controlando la forma en que son anunciadas las rutas BGP. En concreto, X operará como una red terminal si anuncia (a

sus vecinos B y C) que no tiene ninguna ruta a ningún otro destino excepto a ella misma. Es decir, incluso aunque X pueda conocer una determinada ruta, por ejemplo XCY, para llegar a la red Y, *no* anunciará este camino a B. Puesto que B no es consciente de que X dispone de un camino hasta Y, B nunca reenviará tráfico destinado a Y (o a C) a través de X. Este sencillo ejemplo ilustra cómo se puede utilizar una política selectiva de anuncio de rutas para implementar las relaciones de enrutamiento cliente/proveedor.

A continuación, vamos a centrarnos en la red de un proveedor, por ejemplo, en el sistema autónomo B. Suponga que B ha aprendido (de A) que A dispone del camino AW hacia W. B puede por tanto incluir la ruta BAW en su base de información de enrutamiento. Evidentemente, B también quiere anunciar la ruta BAW a su cliente, X, por lo que X sabe que puede llegar a W a través de B. Pero, ¿debería B anunciar la ruta BAW a C? Si lo hace, entonces C podría enviar tráfico a W a través de CBAW. Si A, B y C son proveedores troncales, entonces B podría pensar, con razón, que no tendría que asumir la carga (¡y el coste!) de transportar el tráfico en tránsito entre A y C. B puede pensar justamente que es el trabajo (¡y el coste!) de A y C asegurarse de que C puede enrutar hacia/desde los clientes de A a través de una conexión directa entre A y C. Actualmente, no existe ningún estándar oficial que gobierne cómo los ISP troncales deben llevar a cabo el enrutamiento entre ellos. Sin embargo, una regla heurística seguida por los ISP comerciales es que cualquier tráfico que fluya a través de la red troncal de un ISP tiene que tener su origen o su destino (o ambos) en una red que sea un cliente de dicho ISP; en cualquier otro caso, el tráfico deberá ser expulsado de la red del ISP. Los acuerdos entre pares individuales (que gobernarían cuestiones como las mencionadas más arriba) normalmente son negociados entre parejas de proveedores ISP y suelen ser confidenciales; [Huston 1999a] proporciona una interesante información acerca de los acuerdos entre pares. Si desea ver una descripción detallada de cómo las políticas de enrutamiento reflejan las relaciones comerciales entre los ISP, consulte [Gao 2001; Dmitiropoulos 2007]. Para ver una descripción reciente de las políticas de enrutamiento de BGP desde el punto de vista de un ISP, consulte [Caesar 2005].

Como hemos mencionado anteriormente, BGP es el estándar *de facto* para el enrutamiento entre sistemas autónomos de Internet. Para ver el contenido de varias tablas de enrutamiento BGP (¡muy grandes!) extraídas de routers de los ISP de nivel 1, consulte http://www.routeviews.org. Frecuentemente, las tablas de enrutamiento BGP contienen decenas de miles de prefijos y los atributos correspondientes. Puede ver estadísticas acerca del tamaño y las características de las tablas de enrutamiento BGP en [Huston 2001; Meng 2005; Potaroo 2009].

Con esto terminamos nuestra breve introducción al protocolo BGP. Debe entender que BGP es importante porque desempeña un papel central en Internet. Le animamos a que consulte las siguientes referencias [Griffin 2002; Stewart 1999; Labovitz 1997; Halabi 2000; Huitema 1998; Gao 2001; Feamster 2004; Caesar 2005; Li 2007], con el fin de aprender más sobre BGP.

4.7 Enrutamiento por difusión y por multidifusión

Hasta aquí nos hemos centrado en los protocolos de enrutamiento que soportan las comunicaciones *unicast* o unidifusión (como, por ejemplo, las comunicaciones punto a punto), en las que un único nodo de origen envía un paquete a un único nodo de destino. En esta sección, vamos a ocuparnos de los protocolos de enrutamiento por difusión y por multidifusión.

En el **enrutamiento por difusión**, la capa de red proporciona un servicio de entrega para un paquete enviado desde un nodo de origen a todos los demás nodos de la red; el **enrutamiento por multidifusión** permite a un único nodo de origen enviar una copia de un paquete a un subconjunto de los restantes nodos de la red. En la Sección 4.7.1 abordaremos los algoritmos de enrutamiento por difusión y su plasmación en los protocolos de enrutamiento. En la Sección 4.7.2 examinaremos el enrutamiento por multidifusión.

4.7.1 Algoritmos de enrutamiento por difusión

Quizá la forma más directa de llevar a cabo la comunicación por difusión es que el nodo emisor envíe una copia distinta del paquete a cada destino, como se muestra en la Figura 4.43(a). Dados *N* nodos de destino, el nodo de origen simplemente hace *N* copias del paquete, direcciona cada copia a un destino diferente y luego transmite las *N* copias a los *N* destinos, utilizando el enrutamiento por unidifusión. Este método de **unidifusión por N vías** para llevar a cabo la difusión es simple: no se necesita ningún nuevo protocolo de enrutamiento de la capa de red, ni una duplicación de paquetes ni una funcionalidad de reenvío. Sin embargo, este método presenta varios inconvenientes. El primero de ellos es su ineficiencia. Si el nodo de origen está conectado al resto de la red a través de un único enlace, entonces *N* copias distintas del (mismo) paquete atravesarán ese mismo enlace. Evidentemente, sería más eficiente enviar sólo una única copia del paquete a través del primer salto y que luego el nodo que se encuentra en el otro extremo del primer salto creara y reenviara las copias adicionales necesarias. Es decir, sería más eficiente para los propios nodos de la red (en lugar de sólo para el nodo de origen) crear copias duplicadas de un paquete. Por ejemplo, en la Figura 4.43(b) sólo una única copia de un paquete atraviesa el enlace R1-R2. Después, dicho paquete se duplica en R2, enviándose una única copia a través de los enlaces R2-R3 y R2-R4.

Los otros inconvenientes de la unidifusión por *N* vías quizá son más sutiles, pero no menos importantes. Una suposición implícita de la unidifusión por *N* vías es que los receptores del paquete de difusión y sus direcciones son conocidas por el emisor. Pero, ¿como se ha obtenido esta información? Muy probablemente, serán necesarios mecanismos de protocolo adicionales (tales como un protocolo de pertenencia al dominio de difusión o de registro de destinos). Esto añadiría más sobrecarga y, lo que es más importante, una complejidad adicional a un protocolo que inicialmente parecía bastante simple. Un

Creación/transmisión de duplicado

R1 — Duplicado
R2
R3 R4
a.

R1
R2 — Duplicado
R3 R4
b.

Figura 4.43 • Duplicación en el origen frente a duplicación en la red.

último inconveniente de la unidifusión por *N* vías está relacionado con los propósitos por los que se utiliza la difusión. En la Sección 4.5 vimos que los protocolos de enrutamiento de estado de enlaces utilizan la difusión para diseminar la información de estado de los enlaces que se utiliza para calcular las rutas de unidifusión. Evidentemente, en aquellas situaciones en las que se emplea la difusión para crear y actualizar rutas de unidifusión, no sería prudente (por decir algo) basarse en la infraestructura de enrutamiento por unidifusión para conseguir la comunicación por difusión.

Dados los diversos inconvenientes de la comunicación por difusión utilizando unidifusión por *N* vías, está claro que serán interesantes métodos en los que los propios nodos de la red desempeñen un papel activo en la duplicación y el reenvío de los paquetes y en el cálculo de las rutas de difusión. Vamos a examinar algunos de estos métodos y de nuevo adoptaremos la notación de grafos presentada en la Sección 4.5. Vamos a modelar la red como un grafo, $G = (N,E)$, donde N es un conjunto de nodos y E es una colección de aristas, donde cada arista es un par de nodos de N. Vamos a ser un poco permisivos con la notación y utilizaremos N para hacer referencia tanto al conjunto de nodos como a la cardinalidad ($|N|$) o tamaño de dicho conjunto, cuando no exista ambigüedad.

Inundación no controlada

La técnica más obvia para llevar a cabo la difusión es mediante un método de **inundación** (*flooding*) en el que el nodo de origen envía una copia del paquete a todos sus vecinos. Cuando un nodo recibe un paquete de difusión, lo duplica y lo reenvía a todos sus vecinos (excepto al vecino del que ha recibido el paquete). Evidentemente, si el grafo está conectado, este esquema permitirá terminar entregando una copia del paquete de difusión a todos los nodos del grafo. Aunque este esquema es sencillo y elegante, presenta un error fatal (antes de leerlo, imagine cuál puede ser ese error fatal): si el grafo presenta ciclos, entonces una o más copias de cada paquete de difusión podría estar dando vueltas indefinidamente. Por ejemplo, en la Figura 4.43, R2 inundará a R3, R3 inundará a R4, R4 inundará a R2 y R2 inundará (¡de nuevo!) a R3, y así sucesivamente. Este sencillo escenario da lugar a un bucle sin fin de dos paquetes de difusión, uno en el sentido de las agujas del reloj y otro en sentido contrario. Pero puede darse el caso de un error aún más fatal: cuando un nodo está conectado a más de dos nodos, creará y reenviará múltiples copias del paquete de difusión, que a su vez crearán múltiples copias de sí mismos (en otros nodos con más de dos vecinos), y así sucesivamente. Esta **tormenta de difusión**, que resulta de la multiplicación sin fin de paquetes de difusión, terminaría creando tantos paquetes de difusión que la red quedaría inutilizada. Consulte las cuestiones de repaso al final del capítulo para ver un problema que analiza la velocidad a la que crece una tormenta de difusión.

Inundación controlada

La clave para evitar una tormenta de difusión es que un nodo decida juiciosamente cuándo inundar con un paquete y cuándo (por ejemplo, si ya ha recibido e inundado con una copia anterior del paquete) no realizar tal inundación. En la práctica, esto puede hacerse de varias formas.

En la **inundación controlada por el número de secuencia**, un nodo de origen incluye su dirección (u otro identificador único), así como un **número de secuencia de difusión** en cada paquete de difusión, y a continuación envía el paquete a todos sus vecinos. Cada nodo mantiene una lista de la dirección de origen y el número de secuencia de cada paquete de

difusión que ya ha recibido, duplicado y reenviado. Cuando un nodo recibe un paquete de difusión, primero comprueba si el paquete está en la lista. Si lo está, el paquete se elimina; en caso contrario, duplica y reenvía el paquete a todos sus nodos vecinos (excepto al nodo del que acaba de recibir el paquete). El protocolo Gnutella utiliza la inundación controlada por número de secuencia para difundir las consultas de difusión a lo largo de su red solapada. (En Gnutella, la duplicación y reenvío de mensajes se lleva a cabo en la capa de aplicación en lugar de en la capa de red.)

Un segundo método de inundación controlada es el método conocido como **reenvío por el camino inverso (RPF, *Reverse Path Forwarding*)** [Dalal 1978], en ocasiones también denominado difusión por el camino inverso (RPB, *Reverse Path Broadcast*). La idea que subyace a RPF es simple, e incluso elegante. Cuando un router recibe un paquete de difusión con una determinada dirección de origen, transmite el paquete a todos sus enlaces de salida (excepto al enlace por el que ha sido recibido) sólo si el paquete ha llegado a través del enlace que pertenece a su propia ruta de unidifusión más corta que le conecta con el origen. En otro caso, el router simplemente descarta el paquete entrante sin reenviarlo a ninguno de sus enlaces de salida. Dicho paquete puede ser eliminado porque el router sabe que recibirá o que ya ha recibido una copia de ese paquete a través del enlace que se encuentra en su propia ruta más corta que le conecta con el emisor (puede comprobar usted mismo que esto es así y que no se producirán bucles ni tormentas de difusión.) Observe que RPF no utiliza enrutamiento de unidifusión para entregar realmente un paquete a un destino, ni requiere que un router conozca la ruta completa más corta desde sí mismo hasta el origen. RPF sólo necesita saber cuál es el siguiente vecino en su ruta de unidifusión más corta hacia el emisor; utiliza esta identidad del vecino sólo para determinar si debe inundar o no un paquete de difusión recibido.

La Figura 4.44 ilustra RPF. Suponga que los enlaces dibujados mediante líneas gruesas representan las rutas de coste mínimo desde los receptores al origen (*A*). Inicialmente, el nodo *A* difunde un paquete con origen en *A* a los nodos *C* y *B*. El nodo *B* reenviará el paquete con origen en *A* que ha recibido de *A* (ya que *A* se encuentra en la ruta de coste mínimo hacia *A*) tanto a *C* como a *D*. *B* ignorará (eliminará, sin reenviar) cualquier paquete

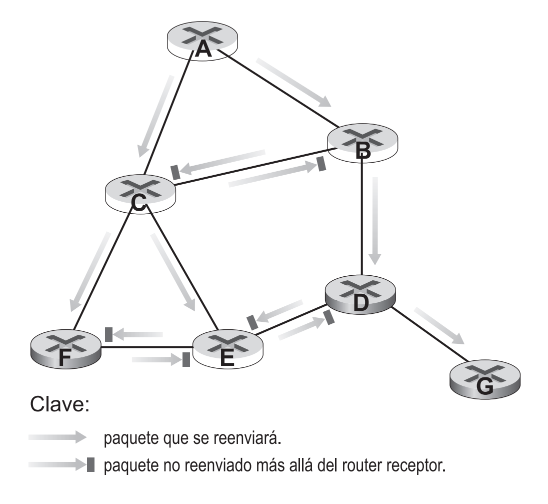


Figura 4.44 • Reenvío por el camino inverso.

cuyo origen sea *A* recibido de cualquier otro nodo (por ejemplo, de los routers *C* o *D*). Consideremos ahora el nodo *C*, el cual recibe un paquete con origen en *A* directamente de *A*, así como de *B*. Puesto que *B* no está en la ruta más corta de *C* a *A*, *C* ignorará los paquetes con origen en *A* recibidos desde *B*. Por el contrario, cuando *C* reciba un paquete con origen en *A* directamente desde *A*, reenviará los paquetes a los nodos *B*, *E* y *F*.

Difusión por árbol de recubrimiento

Aunque la inundación controlada por el número de secuencia y RPF evitan las tormentas de difusión, no evitan completamente la transmisión de paquetes de difusión redundantes. Por ejemplo, en la Figura 4.45, los nodos *B*, *C*, *D*, *E* y *F* reciben uno o dos paquetes redundantes. Idealmente, cada nodo tendría que recibir sólo una copia del paquete de difusión. Examinando el árbol formado por los nodos conectados por las líneas gruesas de la Figura 4.45(a), puede ver que si los paquetes de difusión fueran reenviados sólo por los enlaces de este árbol, todos y cada uno de los nodos de la red recibirían exactamente una copia del paquete de difusión (y ésta es exactamente la solución que estábamos buscando). Este árbol es un ejemplo de un **árbol de recubrimiento (*spanning tree*)** (un árbol que contiene todos y cada uno de los nodos de un grafo). De manera más formal, un árbol de recubrimiento de un grafo $G = (N,E)$ es un grafo $G' = (N,E')$ tal que E' es un subconjunto de E, G' está conectado, G' no contiene ciclos y G' contiene todos los nodos originales de G. Si cada enlace tiene un coste asociado y el coste de un árbol es la suma de los costes de los enlaces, entonces un árbol de recubrimiento cuyo coste sea el mínimo de todos los árboles de recubrimiento del grafo se dice que es (nada sorprendentemente) un **árbol de recubrimiento mínimo**.

Luego otro método que permite la difusión consiste en que los nodos de la red construyan en primer lugar un árbol de recubrimiento. Cuando un nodo de origen desea enviar un paquete de difusión, lo envía a través de todos los enlaces incidentes que pertenecen al árbol de recubrimiento. Un nodo que recibe un paquete de difusión reenvía entonces el paquete a todos sus vecinos del árbol de recubrimiento (excepto al vecino del que ha recibido el paquete). El árbol de recubrimiento no sólo elimina los paquetes de difusión redundantes, sino que una vez que se ha definido, puede ser utilizado por cualquier nodo para iniciar una difusión, como se muestra en las Figuras 4.45(a) y 4.45(b). Observe que un nodo no nece-

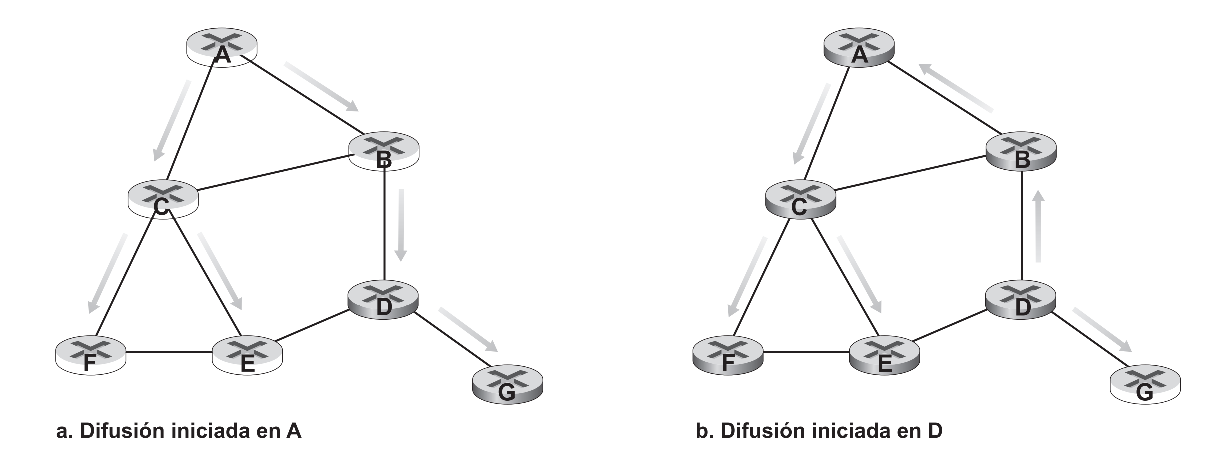
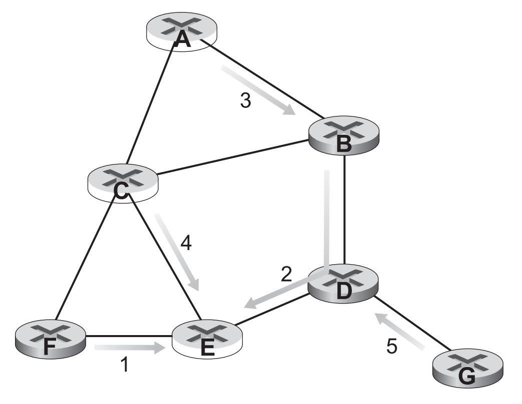


Figura 4.45 • Difusión a lo largo de un árbol de recubrimiento.

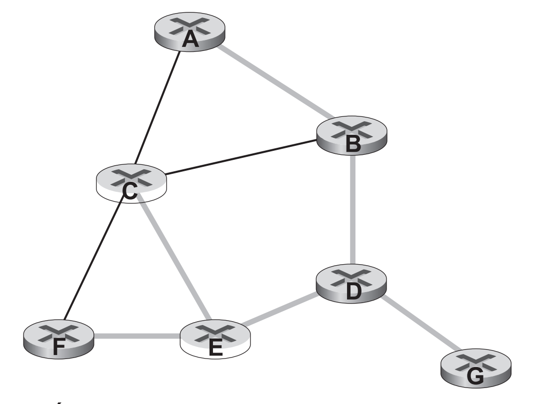
sita ser consciente del árbol completo; simplemente necesita saber cuáles de sus vecinos en *G* son también vecinos dentro del árbol de recubrimiento.

La principal complejidad asociada con el método del árbol de recubrimiento es la creación y mantenimiento del árbol de recubrimiento. Se han desarrollado muchos algoritmos distribuidos de árbol de recubrimiento [Gallager 1983, Gartner 2003]. Aquí sólo vamos a considerar un algoritmo simple. En el **método basado en un nodo central** para construir un árbol de recubrimiento, se define un nodo central (conocido también como un **punto de cita** o ***rendezvous*** o **núcleo**). A continuación, los nodos envían al nodo central mensajes de unidifusión de unión al árbol. Los mensajes de unión al árbol se reenvían hacia el nodo central, utilizando enrutamiento por unidifusión, hasta que llegan a un nodo que ya pertenece al árbol de recubrimiento o hasta que llegan al centro. En ambos casos, la ruta que el mensaje de unión al árbol ha seguido define la rama del árbol de recubrimiento que va desde el centro hasta el nodo arista que inició el mensaje de unión al árbol. En cierto modo, es como si esta nueva ruta hubiera sido agregada al árbol de recubrimiento existente.

La Figura 4.46 ilustra la construcción de un árbol de recubrimiento basado en un nodo central. Suponga que se selecciona el nodo *E* como el centro del árbol. Suponga que el nodo *F* es el primero que se une al árbol y reenvía un mensaje de unión al árbol a *E*. El único enlace *EF* se convierte en el árbol de recubrimiento inicial. El nodo *B* se une a continuación al árbol de recubrimiento enviando a *E* sus mensajes de unión al árbol. Suponga que la ruta de unidifusión a *E* desde *B* es a través de *D*. En este caso, el mensaje de unión al árbol da lugar a la ruta *BDE* que se agrega al árbol de recubrimiento existente. A continuación, el nodo *A* se une al grupo de recubrimiento reenviando su mensaje de unión al árbol hacia *E*. Si la ruta de unidifusión de *A* a *E* es a través de *B*, entonces, puesto que *B* ya está unido al árbol de recubrimiento, la llegada del mensaje de unión al árbol procedente de *A* hacia *B* hará que el enlace *AB* se añada de forma inmediata al árbol de recubrimiento. El nodo *C* se une a continuación al árbol de recubrimiento reenviando su mensaje de unión al árbol directamente a *E*. Finalmente, dado que el enrutamiento por unidifusión de *G* a *E* tiene que hacerse a través de *D*, cuando *G* envía a *E* su mensaje de unión al árbol, se inserta el enlace *GD* en el árbol de recubrimiento, a la altura del nodo *D*.



a. Construcción paso a paso del árbol de recubrimiento.



b. Árbol de recubrimiento reconstruido.

Figura 4.46 • Construcción de un árbol de recubrimiento basada en un nodo central.

Algoritmos de difusión en la práctica

Los protocolos de difusión se utilizan en la práctica tanto en la capa de aplicación como en la de red. Gnutella [Gnutella 2009] utiliza la difusión en el nivel de aplicación para difundir consultas de contenido entre los pares Gnutella. Aquí, cada enlace entre dos pares de procesos distribuidos de nivel de aplicación dentro de la red Gnutella es realmente una conexión TCP. Gnutella utiliza una forma de inundación controlada por número de secuencia en la que se emplean un identificador de 16 bits y un descriptor de carga útil de 16 bits (que identifica el tipo de mensaje de Gnutella) para detectar si una consulta de difusión recibida ha sido recibida, duplicada y reenviada anteriormente. Gnutella también emplea un campo para establecer el tiempo de vida (TTL, *Time-To-Live*) con el fin de limitar el número de saltos a través de los que será reenviada una consulta por inundación. Cuando un proceso Gnutella recibe y duplica una consulta, decrementa el campo TTL antes de reenviar la consulta. Por tanto, una consulta Gnutella por inundación sólo alcanzará a aquellos pares que estén a un número dado (el valor inicial de TTL) de saltos (en el nivel de aplicación) con respecto al iniciador de la consulta. El mecanismo de inundación de Gnutella se denomina a veces *inundación de ámbito limitado.*

También se utiliza una forma de inundación controlada por el número de secuencia para difundir los anuncios de estado de enlaces (LSA, *Link-State Advertisements*) en el algoritmo de enrutamiento OSPF [RFC 2328, Perlman 1999] y en el algoritmo de enrutamiento de sistema intermedio a sistema intermedio (IS-IS, *Intermediate-System-to-Intermediate-System*) [RFC 1142, Perlman 1999]. OSPF utiliza un número de secuencia de 32 bits así como un campo de antigüedad de 16 bits para identificar los anuncios LSA. Recuerde que un nodo OSPF utiliza la comunicación por difusión para enviar periódicamente los LSA a sus enlaces conectados, cuando el coste de un enlace a un vecino cambia o cuando un enlace pasa al estado activo o inactivo. Los números de secuencia de los LSA se emplean para detectar anuncios LSA duplicados, pero también sirven a una segunda función importante en OSPF. Con la inundación, es posible que un LSA generado por el origen en el instante t llegue *después* que un LSA más reciente que haya sido generado por el mismo origen en el instante $t +$. Los números de secuencia utilizados por el nodo de origen permiten diferenciar un LSA más antiguo de un LSA más reciente. El campo antigüedad sirve para un propósito similar al que sirve un valor de TTL. El valor inicial del campo antigüedad se pone a cero y se incrementa en cada salto a medida que tiene lugar la inundación y también se incrementa mientras se encuentra retenido en la memoria de un router, esperando a ser reenviado. Aunque aquí hemos descrito brevemente el algoritmo de inundación de los LSA, de hecho el diseño de los protocolos de difusión LSA puede resultar muy complicado. [RFC 789; Perlman 1999] describen un incidente en el que los anuncios LSA incorrectamente transmitidos por dos routers que funcionaban mal dieron lugar a que una versión temprana de un algoritmo de inundación de anuncios LSA provocara la caída de toda la red ARPAnet.

4.7.2 Multidifusión

Hemos visto en la sección anterior que con un servicio de difusión los paquetes son suministrados a todos y cada uno de los nodos de la red. En esta sección vamos a ocuparnos del servicio de **multidifusión (*multicast*)**, en el que un paquete de multidifusión se entrega a sólo un *subconjunto* de los nodos de la red. Una serie de aplicaciones de red emergentes requiere la entrega de paquetes procedentes de uno o más emisores a un grupo de recepto-

res. Estas aplicaciones incluyen la transferencia masiva de datos (por ejemplo, la transferencia de una actualización software desde el desarrollador a los usuarios que precisan esa actualización), los flujos multimedia continuos (por ejemplo, la transferencia de audio, vídeo y texto de una conferencia en vivo a un conjunto de participantes en la conferencia geográficamente distribuidos), las aplicaciones de datos compartidos (como una aplicación de teleconferencia o de pizarra electrónica que se comparte entre muchos participantes distribuidos), los alimentadores de datos (por ejemplo, las cotizaciones de bolsa), las actualizaciones de caché Web y los juegos interactivos (por ejemplo, los entornos virtuales interactivos distribuidos o los juegos multijugador).

En la mutlidifusión, nos enfrentamos a dos problemas: cómo identificar a los receptores de un paquete de multidifusión y cómo dirigir un paquete enviado a esos receptores. En el caso de la unidifusión, la dirección IP del receptor (el destino) se transporta en cada datagrama IP de unidifusión e identifica a un único receptor; en el caso de la comunicación por difusión, *todos* los nodos tienen que recibir el paquete de difusión, por lo que no se necesita ninguna dirección de destino. Pero en el caso de la multidifusión, tenemos varios receptores. ¿Tiene sentido que cada paquete de multidifusión transporte las direcciones IP de todos esos receptores? Aunque este método podría ser adecuado para un número pequeño de receptores, no se escalaría bien para el caso de cientos o miles de receptores; la cantidad de información de direccionamiento incluida en el datagrama eclipsaría a los datos reales transportados en el campo de carga útil del paquete. Que el emisor haga una identificación explícita de los receptores también requiere que conozca las identidades y direcciones de todos los receptores. Veremos en breve que existen casos en los que este requisito puede no ser deseable.

Por estas razones, en la arquitectura de Internet (y en otras arquitecturas como ATM [Black 1995]), un paquete de multidifusión se direcciona utilizando la **indirección de direcciones**. Es decir, se utiliza un único identificador para el grupo de receptores y a cada uno de los receptores multidifusión asociados con dicho grupo se le entrega una copia del paquete que se ha dirigido al grupo utilizando ese identificador. En Internet, el identificador único que representa a un grupo de receptores es una dirección IP de multidifusión de clase D. El grupo de receptores asociados con una dirección de clase D se conoce como **grupo de multidifusión**. La abstracción de grupo de multidifusión se ilustra en la Figura 4.47. En este ejemplo, cuatro hosts (de los que entran y salen las flechas) están asociados con la dirección del grupo de multidifusión 226.17.30.197 y recibirán todos los datagramas dirigidos a esa dirección de multidifusión. La dificultad que todavía tenemos que superar es que cada host tiene una dirección IP de unidifusión única que es completamente independiente de la dirección del grupo de multidifusión en el que participa.

Aunque la abstracción de grupo de multidifusión es simple, plantea un montón de preguntas. ¿Cómo se inicia y se termina un grupo? ¿Cómo se elige la dirección de grupo? ¿Cómo se añaden nuevos hosts al grupo (sean emisores o receptores)? ¿Puede alguien unirse a un grupo (y enviar al grupo y recibir del grupo) o la pertenencia al grupo está restringida, y si lo está, quién la restringe? ¿Conocen los miembros del grupo las identidades de los otros miembros del grupo como parte del protocolo de la capa de red? ¿Cómo interoperan los nodos de la red entre sí para entregar un datagrama de multidifusión a todos los miembros del grupo? En Internet, las respuestas a todas estas preguntas implican al Protocolo de gestión de grupos de Internet (IGMP, *Internet Group Management Protocol*) [RFC 3376]. Por tanto, a continuación vamos a ver brevemente el protocolo IGMP y luego volveremos sobre todas estas preguntas.

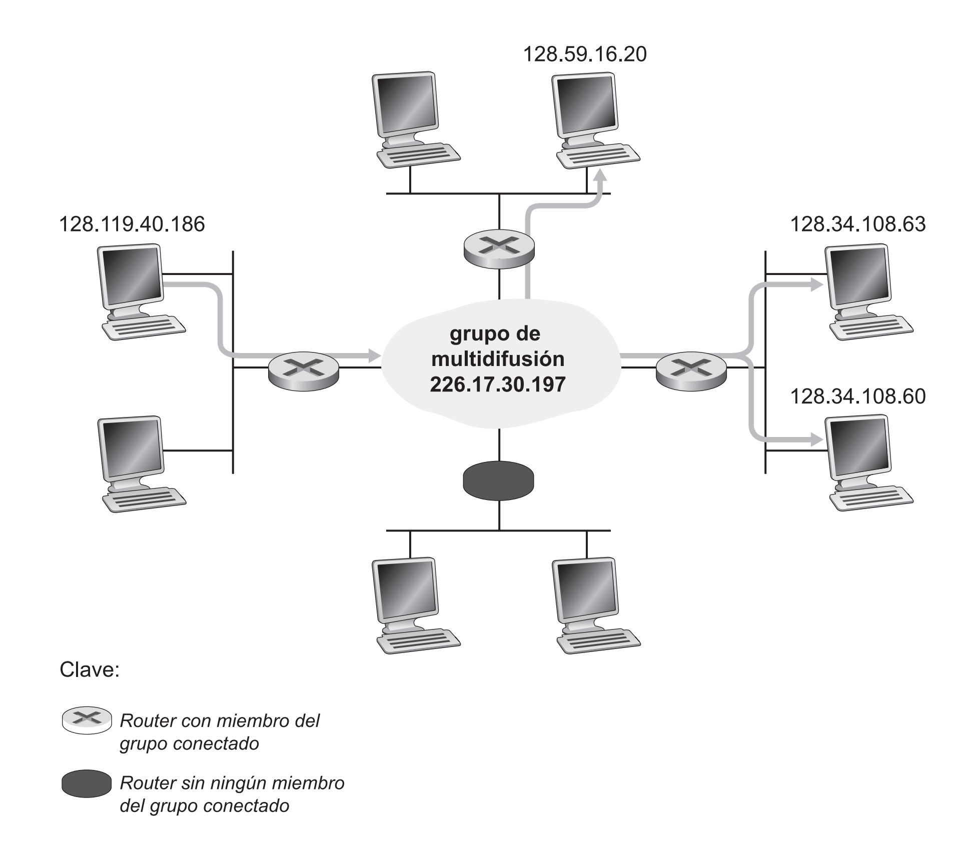


Figura 4.47 • Grupo de multidifusión: un datagrama dirigido al grupo se entrega a todos los miembros del grupo de multidifusión.

Protocolo de gestión de grupos de Internet (IGMP)

La versión 3 del protocolo IGMP [RFC 3376] opera entre un host y su router directamente conectado (informalmente, podemos pensar en el router directamente conectado como en el router del primer salto que un host vería en una ruta hacia cualquier host que se encuentra fuera de su propia red local, o el router del último salto en cualquier ruta a dicho host), como se muestra en la Figura 4.48. La Figura 4.48 muestra tres routers multidifusión de primer salto, cada uno de ellos conectado a sus hosts a través de una interfaz local de salida. En este ejemplo, esta interfaz local está conectada a una LAN, y aunque cada LAN tiene varios hosts conectados, normalmente como máximo sólo unos pocos de estos hosts pertenecerán a un grupo de multidifusión dado en cualquier instante determinado.

IGMP proporciona los medios a un host para informar a su router conectado de que una aplicación que se ejecuta en el host desea unirse a un grupo de multidifusión especificado. Puesto que el ámbito de la interacción IGMP está limitado a un host y al router al que está conectado, está claro que se requiere otro protocolo para coordinar a los routers multidifusión (incluyendo a los routers conectados) a través de Internet, con el fin de enrutar los datagramas de multidifusión hacia sus destinos finales. Esta última funcionalidad se consigue mediante algoritmos de enrutamiento por multidifusión de la capa de red, tales como los que

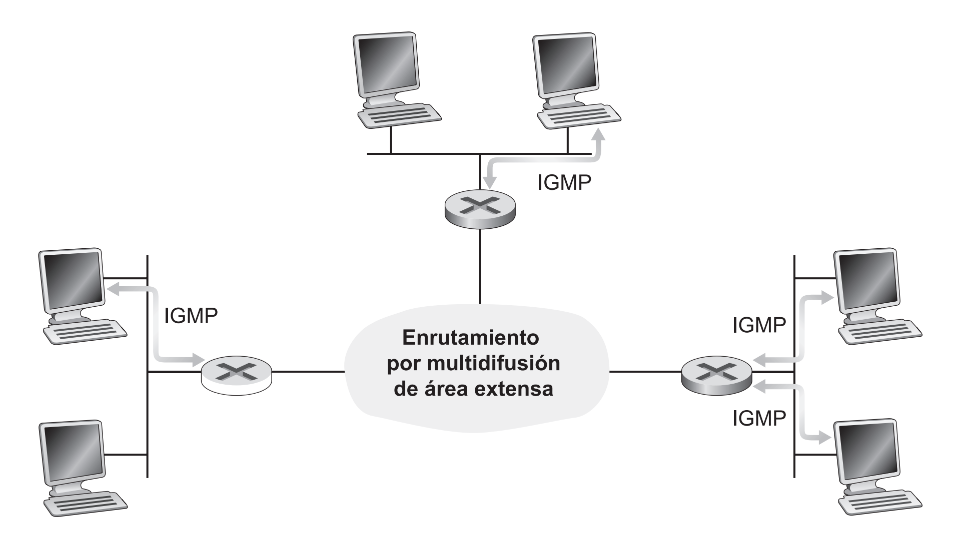


Figura 4.48 • Los dos componentes de la multidifusión de la capa de red en Internet: IGMP y los protocolos de enrutamiento por multidifusión.

veremos a continuación. Por tanto, la multidifusión de la capa de red en Internet tiene dos componentes complementarios: IGMP y los protocolos de enrutamiento por multidifusión.

IGMP sólo dispone de tres tipos de mensajes. Al igual que ICMP, los mensajes IGMP son transportados (encapsulados) dentro de un datagrama IP, con un número de protocolo IP de 2. El mensaje `membership_query` lo envía un router a todos los hosts situados en una interfaz conectada (por ejemplo, a todos los hosts de una red de área local) para determinar el conjunto de todos los grupos de multidifusión a los que han unido los hosts conectados a través de dicha interfaz. Los hosts responden a un mensaje `membership_query` con un mensaje IGMP `membership_report`. Un host también puede generar mensajes `membership_report` cuando una aplicación se une por primera vez a un grupo de multidifusión sin esperar a un mensaje `membership_query` procedente del router. El último tipo de mensaje IGMP es el mensaje `leave_group`. Curiosamente, este mensaje es opcional. Pero aunque sea opcional, ¿cómo detecta un router cuándo un host abandona el grupo de multidifusión? La respuesta a esta pregunta es que el router *infiere* que un host ya no está en el grupo de multidifusión si no responde a un mensaje `membership_query` con la dirección de un determinado grupo. Esto es un ejemplo de lo que en ocasiones se denomina **estado frágil (soft-state)** en un protocolo de Internet. En un protocolo de estado frágil, el estado (en nuestro caso de IGMP, el hecho de que existan hosts unidos a un determinado grupo de multidifusión) desaparece debido a un suceso de fin de temporización (en este caso, mediante un mensaje `membership_query` periódico del router) si no se refresca explícitamente (en este caso, mediante un mensaje `membership_report` procedente de un host conectado). Hay quien dice que los protocolos de estado frágil proporcionan un control más simple que los protocolos de estado firme (*hard-state*), que no sólo requieren que el estado sea explícitamente añadido y eliminado, sino que también necesitan mecanismos para recuperarse de la situación en la que la entidad responsable de eliminar el estado haya terminado de forma prematura o ha fallado. Puede encontrar una interesante exposición sobre el tema del estado frágil en [Raman 1999; Ji 2003; Lui 2004].

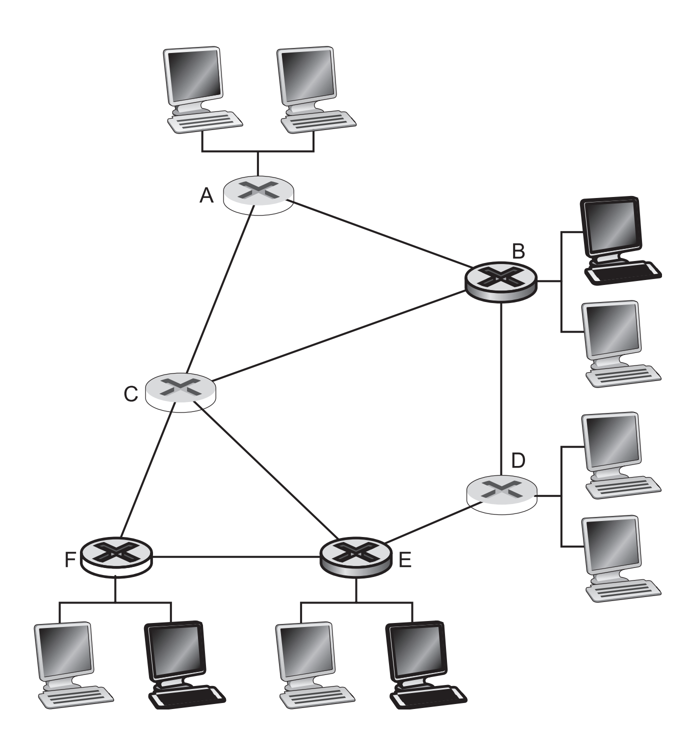


Figura 4.49 • Hosts multidifusión, sus routers conectados y otros routers.

Algoritmo de enrutamiento por multidifusión

En la Figura 4.49 se ilustra el **problema del enrutamiento por multidifusión**. Los hosts unidos al grupo de multidifusión están bordeados en negro, así como su router inmediatamente conectado. Como se muestra en la Figura 4.49, sólo un subconjunto de routers (aquellos con hosts conectados que están unidos al grupo de multidifusión) necesita realmente recibir el tráfico multidifusión. En la Figura 4.49, sólo los routers *A*, *B*, *E* y *F* tienen que recibir el tráfico de multidifusión. Dado que ninguno de los hosts conectados al router *D* está unido al grupo de multidifusión y puesto que el router *C* no tiene hosts conectados, ni *C* ni *D* necesitan recibir el tráfico del grupo de multidifusión. El objetivo del enrutamiento por mutlidifusión es, por tanto, encontrar un árbol de enlaces que conecte todos los routers que tienen hosts conectados que pertenecen al grupo de multidifusión. Los paquetes de multidifusión entonces serán enrutados a lo largo de este árbol desde el emisor a todos los hosts que pertenecen al árbol de multidifusión. Por supuesto, el árbol puede contener routers que no tengan hosts conectados que pertenezcan al grupo de multidifusión (por ejemplo, en la Figura 4.49 es imposible conectar los routers *A*, *B*, *E* y *F* en un árbol sin implicar bien al router *C* o al router *D*).

En la práctica, se han adoptado dos métodos para determinar el árbol de enrutamiento por multidifusión, que ya hemos estudiado en el contexto del enrutamiento por difusión y, por tanto, aquí sólo lo vamos a mencionar de pasada. Los dos métodos difieren dependiendo de si se utiliza un único árbol compartido por el grupo para distribuir el tráfico a *todos* los emisores del grupo, o si se construye un árbol de enrutamiento específico del origen para cada emisor individual.

- *Enrutamiento por multidifusión que utiliza un árbol compartido por el grupo.* Como en el caso de la difusión mediante árbol de recubrimiento, el enrutamiento por multidifusión sobre un árbol compartido por el grupo está basado en la construcción de un árbol que incluye a todos los routers de frontera con hosts conectados que pertenecen al grupo de multidifusión. En la práctica, se utiliza el método basado en un nodo central para construir el árbol de enrutamiento por multidifusión, encargándose aquellos routers de frontera que tienen hosts conectados pertenecientes al grupo de multidifusión de enviar (vía unidifusión) mensajes de unión al árbol dirigidos al nodo central. Como en el caso de la difusión, un mensaje de unión se reenvía utilizando el enrutamiento por unidifusión hacia el nodo central, hasta que o bien llega a un router que ya pertenece al árbol de multidifusión o bien llega al nodo central. Todos los routers existentes a lo largo de la ruta que sigue el mensaje de unión reenviarán entonces los paquetes de multidifusión recibidos al router de frontera que inició la unión al grupo de multidifusión. Una cuestión crítica en el enrutamiento por multidifusión mediante árbol basado en un nodo central es el proceso utilizado para seleccionar el nodo central. Puede ver una serie de algoritmos de selección del nodo central en [Wall 1980; Thaler 1997; Estrin 1997].
- *Enrutamiento por multidifusión utilizando un árbol basado en el origen*. Mientras que el enrutamiento por multidifusión basado en árbol compartido por el grupo construye un único árbol de enrutamiento compartido para enrutar los paquetes de *todos* los emisores, el segundo método construye un árbol de enrutamiento por multidifusión para *cada* origen existente en el grupo multidifusión. En la práctica, se utiliza un algoritmo RPF (con *x* como nodo de origen) para construir un árbol de reenvío por multidifusión para los

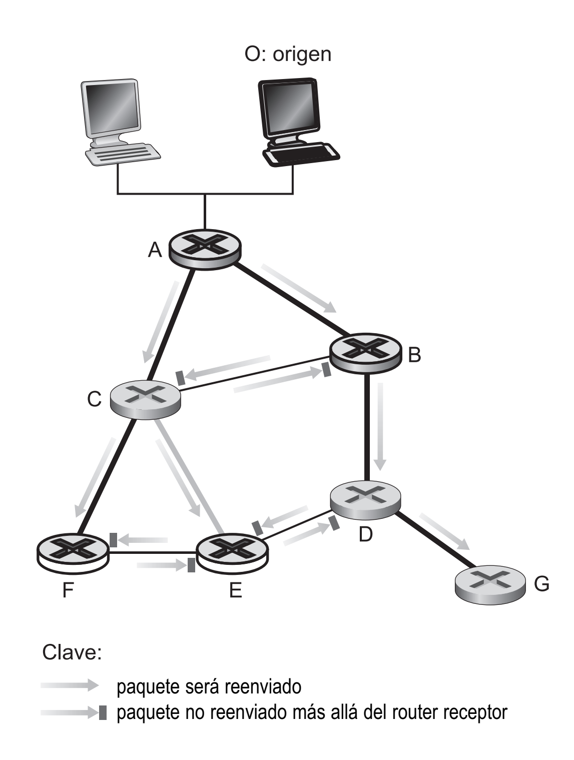


Figura 4.50 • Reenvío de camino inverso para el caso de multidifusión.

datagramas de multidifusión que tienen su origen en *x*. El algoritmo RPF de difusión que hemos estudiado anteriormente requiere ciertos ajustes para utilizarlo en multidifusión. Veamos por qué. Considere el router *D* de la Figura 4.50. Con el algoritmo RPF de difusión, los paquetes se reenviarían al router *G*, incluso aunque el router *G* no tenga hosts conectados que estén unidos al grupo de multidifusión. Aunque esto no es tan malo para este caso en que sólo hay un router (*G*) por debajo de *D*, imagine lo que ocurriría si *D* tuviera miles de routers colgando de él en el árbol. Cada uno de esos miles de routers recibiría paquetes de multidifusión no deseados. (Este escenario no es tan inverosímil como pueda parecer. La MBone inicial [Casner 1992; Macedonia 1994], la primera red de multidifusión global, sufrió precisamente este problema en sus inicios.) La solución al problema de la recepción de paquetes de multidifusión no deseados con RPF se conoce con el nombre de **poda (*pruning*)**. Un router multidifusión que recibe paquetes de multidifusión y que no tiene hosts conectados unidos a dicho grupo enviará un mensaje de poda a su router inmediatamente anterior. Si un router recibe mensajes de poda procedentes de cada uno de los routers situados por debajo, puede entonces reenviar hacia arriba un mensaje de poda.

Enrutamiento por multidifusión en Internet

El primer protocolo de enrutamiento por multidifusión utilizado en Internet fue el **Protocolo de enrutamiento por multidifusión por vector de distancias (DVMRP, *Distance-Vector Multicast Routing Protocol*)** [RFC 1075]. DVMRP implementa árboles basados en el origen, con reenvío de camino inverso y poda. DVMRP utiliza un algoritmo RPF con poda, como hemos visto anteriormente. Quizá el protocolo de enrutamiento por multidifusión de Internet más ampliamente utilizado sea el **protocolo de enrutamiento PIM (*Protocol-Independent Multicast*, Multidifusión independiente del protocolo)**, que reconoce de forma explícita dos escenarios de distribución multidifusión. En el modo denso [RFC 3973], los miembros del grupo de multidifusión están localizados de forma densa, es decir, muchos o la mayoría de los routers del área necesitan involucrarse en el enrutamiento de los datagramas de multidifusión. El modo denso del protocolo PIM es una técnica de reenvío de camino inverso con inundación y poda, similar en espíritu a DVMRP.

En el modo disperso [RFC 4601], el número de routers con miembros del grupo conectados es menor con respecto al número total de routers; los miembros del grupo están tremendamente dispersos. El modo disperso de PIM utiliza puntos de cita para configurar el árbol de distribución de multidifusión. En la **multidifusión específica del origen (SSM, *Source-Specific Multicast*)** [RFC 3569, RFC 4607], sólo un emisor puede enviar tráfico al árbol de multidifusión, simplificando considerablemente la construcción y el mantenimiento del árbol.

Cuando se emplean los protocolos PIM y DVMP dentro de un dominio, el operador de la red puede configurar routers multidifusión IP dentro del dominio, de la misma forma que pueden configurarse protocolos de enrutamiento por unidifusión internos al dominio, como RIP, IS-IS y OSPF. Pero, ¿qué ocurre cuando se necesitan rutas de multidifusión entre los distintos dominios? ¿Existe un equivalente multidifusión del protocolo BGP entre dominios? La respuesta es (literalmente) sí. [RFC 4271] define extensiones multiprotocolo a BGP para permitirle transportar información de enrutamiento para otros protocolos, incluyendo información de multidifusión. El Protocolo de descubrimiento de origen de multidifusión (MSDP, *Multicast Source Discovery Protocol*) [RFC 3618, RFC 4611] se puede utilizar para conectar puntos de cita situados en diferentes dominios PIM en modo disperso. Puede

encontrar una excelente introducción al estado actual del enrutamiento por multidifusión en Internet en [RFC 5110].

Terminamos esta exposición indicando que a la comunicación IP por multidifusión todavía le queda un largo camino por recorrer. Si desea ver una interesante exposición acerca de los problemas de implantación y del modelo de servicio de multidifusión de Internet actual, consulte [Diot 2000, Sharma 2003]. No obstante, a pesar de la falta de una amplia implantación, la tecnología de multidifusión de nivel de red no está "muerta" en absoluto. El tráfico de multidifusión ha estado siendo transportado durante muchos años sobre Internet 2 y sobre las redes equivalentes a ella [Internet2 Multicast 2009]. En el Reino Unido, la BBC participa en pruebas de distribución de contenido mediante IP multidifusión [BBC Multicast 2009]. Al mismo tiempo, la multidifusión en el nivel de aplicación, como hemos visto con PPLive en el Capítulo 2 y en otros sistemas entre pares como *End System Multicast* [ESM 2007], proporcionan distribución por multidifusión de contenido entre pares utilizando protocolos de multidifusión de la capa de aplicación (en lugar de protocolos de multidifusión de la capa de red). ¿Los servicios de multidifusión futuros se implementarán principalmente en la capa de red (en el núcleo de la red) o en la capa de aplicación (en la frontera de la red)? Aunque el entusiasmo actual por la distribución de contenido utilizando protocolos de comunicación entre pares parece inclinar la balanza en favor de las técnicas de multidifusión de la capa de aplicación, al menos durante el futuro más cercano, continúan realizándose avances en la tecnología de multidifusión IP y, en ocasiones, las carreras las ganan los que son más lentos, pero más constantes.

4.8 Resumen

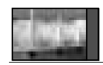
En este capítulo hemos iniciado nuestro viaje al núcleo de la red. Hemos visto que la capa de red implica a todos y cada uno de los hosts y routers de una red. En consecuencia, los protocolos de la capa de red se encuentran entre los más complejos de la pila de protocolos.

Hemos visto que un router puede tener que procesar millones de flujos de paquetes entre distintas parejas origen-destino a un mismo tiempo. Para que un router pueda procesar tal cantidad de flujos, los diseñadores de redes han ido aprendiendo a lo largo de los años que las tareas que debe realizar un router tienen que ser tan simples como sea posible. Se pueden tomar muchas medidas para facilitar el trabajo a los routers, incluyendo el uso de una capa de red de datagramas en lugar de una capa de red de circuitos virtuales; la utilización de una cabecera simplificada y de tamaño fijo (como en IPv6), la eliminación de la fragmentación (también en IPv6) y la provisión del famoso servicio de mejor esfuerzo. Quizá aquí lo más importante *no* es controlar los flujos individuales, sino basar las decisiones de enrutamiento únicamente en las direcciones de destino estructuradas de forma jerárquica de los datagramas. Es interesante destacar que el servicio de correos ordinario ha estado empleando este método durante muchos años.

En este capítulo también hemos examinado los principios en que se basan los algoritmos de enrutamiento. Hemos visto cómo estos algoritmos representan la red de computadoras mediante un grafo con nodos y enlaces. Con esta abstracción, podemos aprovechar la rica teoría de cálculo de la ruta mínima en los grafos, que ha sido desarrollada a lo largo de los últimos 40 años por la comunidad científica dedicada a los algoritmos y la investigación de operaciones. Hemos visto que existen dos métodos generales: un método centralizado (global), en el que cada nodo obtiene un mapa completo de la red y aplica de forma indepen-

diente un algoritmo de enrutamiento basado en la ruta más corta; y un método descentralizado, en el que los nodos individuales sólo disponen de una imagen parcial de la red completa, y los nodos trabajan conjuntamente para entregar paquetes a lo largo de las rutas más cortas. También hemos estudiado cómo se utiliza la jerarquía para afrontar el problema del escalado, dividiendo las redes de gran tamaño en dominios administrativos independientes conocidos como sistemas autónomos (AS). Cada AS enruta de forma independiente sus datagramas a través del sistema autónomo, al igual que cada nación distribuye de forma independiente su correo postal a través del país. Hemos examinado cómo los métodos centralizado, descentralizado y jerárquico se integran en los principales protocolos de enrutamiento de Internet: RIP, OSPF y BGP. Hemos concluido nuestro estudio sobre los algoritmos de enrutamiento considerando los enrutamientos por difusión y por multidifusión.

Una vez completado nuestro estudio sobre la capa de red, vamos a descender un escalón más por la pila de protocolos hasta la capa de enlace. Al igual que la capa de red, la capa de enlace también forma parte del núcleo de la red. Pero en el siguiente capítulo veremos que la capa de enlace tiene la tarea mucho más localizada de transferir los paquetes entre nodos situados en un mismo enlace o LAN. Aunque esta tarea puede parecer en principio trivial comparada con las tareas que desempeña la capa de red, veremos que la capa de enlace implica una serie de cuestiones importantes y fascinantes que nos mantendrán ocupados durante bastante tiempo.



Problemas y cuestiones de repaso

Capítulo 4 Cuestiones de repaso

SECCIONES 4.1–4.2

R1. Revisemos parte de la terminología utilizada en el libro. Recuerde que el nombre que recibe un paquete de la capa de transporte es *segmento* y que el nombre de un paquete de la capa de enlace es *trama*. ¿Cuál es el nombre de un paquete de la capa de red? Recuerde que tanto los routers como los dispositivos de conmutación de la capa de enlace se denominan *conmutadores de paquetes*. ¿Cuál es la diferencia fundamental entre un router y un dispositivo de conmutación de la capa de enlace? Recuerde que utilizamos el término *routers* tanto para las redes de datagramas como para las redes de circuitos virtuales.

R2. ¿Cuáles son las dos funciones más importantes de la capa de red en una red de datagramas? ¿Cuáles son las tres funciones más importantes de la capa de red en una red de circuitos virtuales?

R3. ¿Cuál es la diferencia entre enrutamiento y reenvío?

R4. ¿Utilizan los routers en las redes de datagramas y de circuitos virtuales tablas de reenvío? En caso afirmativo, describa las tablas de reenvío para ambas clases de redes.

R5. Describa algunos servicios hipotéticos que la capa de red pueda proporcionar a un cierto paquete. Haga lo mismo para un flujo de paquetes. ¿Algunos de sus hipotéticos

servicios pueden ser proporcionados por la capa de red de Internet? ¿Alguno es proporcionado por el modelo de servicio CBR de las redes ATM? ¿Alguno es proporcionado por el modelo de servicio ABR de las redes ATM?

R6. Enumere algunas aplicaciones que podrían beneficiarse del modelo de servicio CBR de las redes ATM.

SECCIÓN 4.3

R7. Explique por qué cada puerto de entrada de un router de alta velocidad almacena una copia de la tabla de reenvío.

R8. En la Sección 4.3 se han abordado tres tipos de entramados de conmutación. Enumere y describa brevemente cada uno de ellos.

R9. Describa cómo pueden perderse paquetes en los puertos de entrada. Describa cómo puede eliminarse la pérdida de paquetes en los puertos de entrada (sin utilizar buffers de capacidad infinita).

R10. Describa cómo puede producirse una pérdida de paquetes en los puertos de salida.

R11. ¿Qué es el bloqueo HOL? ¿Se produce en los puertos de entrada o en los puertos de salida?

SECCIÓN 4.4

R12. ¿Tienen direcciones IP los routers? En caso afirmativo, ¿cuántas?

R13. ¿Cuál es el equivalente binario de 32 bits de la dirección IP 223.1.3.27?

R14. Visite un host que utilice DHCP para obtener su dirección IP, su máscara de red, su router predeterminado y la dirección IP de su servidor DNS local. Enumere estos valores.

R15. Suponga que hay tres routers entre un host de origen y un host de destino. Ignorando la fragmentación, se envía un datagrama IP desde el host de origen al host de destino. ¿A través de cuántas interfaces pasará? ¿Cuántas tablas de reenvío indexará para transportar el datagrama desde el origen al destino?

R16. Suponga una aplicación que genera fragmentos de 40 bytes de datos cada 20 milisegundos y cada fragmento se encapsula en un segmento TCP y luego en un datagrama IP. ¿Qué porcentaje de cada datagrama será información administrativa y qué porcentaje será datos de aplicación?

R17. Suponga que el host A envía al host B un segmento TCP encapsulado en un datagrama IP. Cuando el host B recibe el datagrama, ¿cómo sabe la capa de red del host B que debería pasar el segmento (es decir, la carga útil del datagrama) a TCP en lugar de a UDP o a cualquier otro protocolo?

R18. Suponga que adquiere un router inalámbrico y que lo conecta a su módem por cable. Suponga también que su ISP asigna dinámicamente una dirección IP a su dispositivo conectado (es decir, a su router inalámbrico). Además, suponga que tiene cinco equipos PC en su domicilio que utilizan 802.11 para conectarse de forma inalámbrica a su

router inalámbrico. ¿Cómo se asignan las direcciones IP a los cinco PC? ¿Utiliza NAT el router inalámbrico? ¿Por qué?

R19. Compare y contraste los campos de cabecera de IPv4 e IPv6. ¿Tienen campos en común?

R20. A veces se dice que cuando IPv6 tuneliza a través de los routers IPv4, IPv6 trata los túneles de IPv4 como protocolos de la capa de enlace. ¿Está de acuerdo con esta afirmación? ¿Por qué?

SECCIÓN 4.5

R21. Compare y contraste los algoritmos de enrutamiento de estado de enlaces y de vector de distancias.

R22. Explique cómo la organización jerárquica de Internet ha hecho posible el escalar la red a millones de usuarios.

R23. ¿Es necesario que todos los sistemas autónomos utilicen el mismo algoritmo de enrutamiento interno? ¿Por qué?

SECCIÓN 4.6

R24. Considere la Figura 4.37. Comenzando con la tabla original en *D,* suponga que *D* recibe de *A* el siguiente anuncio:

Subred de destino	Siguiente router	Número de saltos al destino
z	C	10
w	—	1
x	—	1
....	....	....

¿Cambiará la tabla en el router *D*? En caso afirmativo, ¿cómo?

R25. Compare y contraste los anuncios utilizados por RIP y OSPF.

R26. Rellene el espacio en blanco. Los anuncios RIP normalmente anuncian el número de saltos a varios destinos. Por el contrario, las actualizaciones BGP anuncian __________ a los distintos destinos.

R27. ¿Por qué es diferente el protocolo de enrutamiento interno de un AS del protocolo de enrutamiento entre sistemas autónomos de Internet?

R28. ¿Por qué no son tan importantes las consideraciones de políticas en los protocolos de enrutamiento internos de los sistemas autónomos, como OSPF y RIP, como lo son para el protocolo de enrutamiento entre sistemas autónomos BGP?

R29. Defina y contraste los siguientes términos: *subred*, *prefijo* y *ruta BGP*.

R30. ¿Cómo utiliza BGP el atributo NEXT-HOP? ¿Cómo utiliza el atributo AS-PATH?

R31. Describa cómo un administrador de red de un ISP de nivel superior puede implementar ciertas políticas al configurar BGP.

SECCIÓN 4.7

R32. ¿Cuál es una diferencia importante entre la implementación de la abstracción de la difusión mediante varias comunicaciones por unidifusión y una única red (router) que soporte difusión?

R33. Para cada uno de los tres métodos generales que hemos estudiado para la difusión (inundación no controlada, inundación controlada y mediante árbol de recubrimiento) indique si las siguientes afirmaciones son verdaderas o falsas. Puede suponer que no se pierde ningún paquete por desbordamiento del buffer y que todos los paquetes son entregados a través de un enlace en el mismo orden en que fueron enviados.

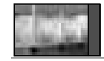
a. Un nodo puede recibir varias copias del mismo paquete.

b. Un nodo puede reenviar múltiples copias de un paquete a través del mismo enlace de salida.

R34. Cuando un host se une a un grupo de multidifusión, ¿tiene que cambiar su dirección IP a la del grupo de multidifusión al que se está uniendo?

R35. ¿Cuáles son las funciones desempeñadas por el protocolo IGMP y por un protocolo de enrutamiento por multidifusión de área extensa?

R36. ¿Cuál es la diferencia entre un árbol compartido por el grupo y un árbol basado en un origen, en el contexto del enrutamiento por multidifusión?



Problemas

P1. En este problema vamos a considerar algunas de las ventajas y de los inconvenientes de las redes de circuitos virtuales y de datagramas.

a. Suponga que los routers están sujetos a condiciones que pueden hacer que fallen con frecuencia. ¿Aconsejaría esto una arquitectura de datagramas o una de circuitos virtuales? ¿Por qué?

b. Suponga que un nodo de origen y un nodo de destino requieren que una cierta capacidad fija esté siempre disponible en todos los routers de la ruta entre el origen y el destino, para el uso exclusivo del tráfico que fluye entre dichos nodos de origen y de destino. ¿Aconsejaría esto una arquitectura de datagramas o una de circuitos virtuales? ¿Por qué?

c. Suponga que los enlaces y routers de la red nunca fallan y que los caminos de enrutamiento utilizados entre todas las parejas origen-destino nunca varían. En este escenario, ¿qué arquitectura (de datagramas o de circuitos virtuales) tiene más sobrecarga de tráfico de control? ¿Por qué?

P2. Considere una red de circuitos virtuales. Suponga que el número de VC es un campo de 8 bits.

a. ¿Cuál es el número máximo de circuitos virtuales que pueden ser transportados a través de un enlace?

b. Suponga que un nodo central determina las rutas y números de VC durante la configuración de la conexión. Suponga que se emplea el mismo número de VC en cada

enlace a lo largo del camino de circuitos virtuales. Describa cómo puede el nodo central determinar el número de VC durante la configuración de la conexión. ¿Es posible que haya menos circuitos virtuales activos que el máximo determinado en el apartado (a) y que, a pesar de ello, no exista ningún número VC común libre?

c. Suponga que están permitidos diferentes números de VC en cada uno de los enlaces que forman el camino de un circuito virtual. Durante la configuración de la conexión, después de que se ha determinado un camino terminal a terminal, describa cómo pueden los enlaces elegir sus números de VC y configurar sus tablas de reenvío de una forma descentralizada, sin basarse en un nodo central.

P3. Una tabla de reenvío mínima en una red de circuitos virtuales tiene cuatro columnas. ¿Cuál es el significado de los valores de cada una de las columnas? Una tabla de reenvío mínima en una red de datagramas tiene dos columnas. ¿Cuál es el significado de los valores de cada una de estas columnas?

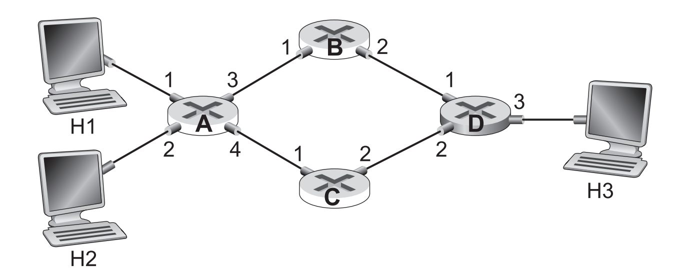
P4. Utilice la red mostrada más abajo.

a. Suponga que se trata de una red de datagramas. Especifique la tabla de reenvío del router A, de modo que todo el tráfico destinado al host H3 sea reenvíado a través de la interfaz 3.

b. Suponga que se trata de una red de datagramas. ¿Puede escribir una tabla de reenvío para el router A, de manera que todo el tráfico de H1 destinado al host H3 sea reenviado a través de la interfaz 3, mientras todo el tráfico de H2 destinado al host H3 sea reenviado a través de la interfaz 4? (*Sugerencia*: esta pregunta tiene truco.)

c. Ahora suponga que se trata de una red de circuitos virtuales y que hay una llamada activa entre H1 y H3, y otra llamada activa entre H2 y H3. Escriba la tabla de reenvío del router A, de modo que todo el tráfico de H1 destinado al host H3 sea reenviado a través de la interfaz 3, mientras todo el tráfico de H2 destinado al host H3 sea reenviado a través de la interfaz 4.

d. Suponga el mismo escenario que en el apartado (c) y escriba las tablas de reenvío de los nodos B, C y D.



P5. Considere una red de circuitos virtuales con un campo de 2 bits para el número de VC. Suponga que la red desea configurar un circuito virtual a través de cuatro enlaces: el enlace A, el enlace B, el enlace C y el enlace D. Suponga también que cada uno de estos enlaces actualmente está dando soporte a otros dos circuitos virtuales y que los números de VC de los mismos son los siguientes:

Enlace A	Enlace B	Enlace C	Enlace D
00	01	10	11
01	10	11	00

Al responder a las siguientes preguntas, tenga en cuenta que cada uno de los circuitos virtuales existentes sólo puede estar atravesando uno de los cuatro enlaces:

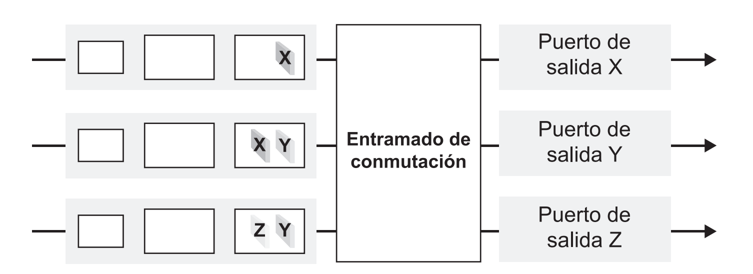
a. Si se requiere que cada circuito virtual utilice el mismo número de VC en todos los enlaces a lo largo de su ruta, ¿qué número de VC podría asignarse al nuevo circuito virtual?

b. Si se permite que cada circuito virtual tenga números de VC distintos en los diferentes enlaces a lo largo de su ruta (de modo que las tablas de reenvío tengan que llevar a cabo una traducción de número de VC), ¿cuántas combinaciones distintas de cuatro números de VC (una para cada uno de los cuatro enlaces) podrían utilizarse?

P6. En el texto hemos utilizado los términos "servicio orientado a la conexión" para describir un servicio de la capa de transporte y "servicio de conexión" para un servicio de la capa de red. ¿Por qué esas sutiles diferencias en la terminología?

P7. En la Sección 4.3 hemos mencionado que puede no existir una cola de entrada si el entramado de conmutación es *n* veces más rápido que las velocidades de línea de entrada, suponiendo que las *n* líneas de entrada presentan todas la misma velocidad de línea. Explique por qué esto debería ser así.

P8. Considere el dispositivo de conmutación mostrado más abajo. Suponga que todos los datagramas tienen la misma longitud fija, que el dispositivo opera de forma síncrona y particionada y que en una partición temporal se puede transferir un datagrama desde un puerto de entrada a un puerto de salida. El entramado de conmutación emplea una estructura de malla, por lo que, como máximo, se puede transferir un datagrama a un puerto de salida determinado en una partición de tiempo, pero distintos puertos de salida pueden recibir datagramas procedentes de distintos puertos de entrada en una misma partición de tiempo. ¿Cuál es el número mínimo de particiones de tiempo necesarias para transferir los paquetes mostrados desde los puertos de entrada a sus puertos de salida, suponiendo cualquier orden de planificación de la cola de entrada que desee (es decir, no tiene por qué existir el bloqueo HOL)? ¿Cuál es el número máximo necesario de particiones de tiempo, suponiendo el orden de planificación de caso peor que pueda imaginar y suponiendo que una cola de entrada no vacía nunca está inactiva?



P9. Considere una red de datagramas que utiliza direcciones de host de 32 bits. Suponga que un router tiene cuatro enlaces, numerados de 0 a 3 y que los paquetes son reenviados a las interfaces de los enlaces como sigue:

Rango de direcciones de destino	Interfaz de enlace
11100000 00000000 00000000 00000000 hasta 11100000 00111111 11111111 11111111	0
11100000 01000000 00000000 00000000 hasta 11100000 01000000 11111111 11111111	1
11100000 01000001 00000000 00000000 hasta 11100001 01111111 11111111 11111111	2
en otro caso	3

a. Proporcione una tabla de reenvío con cuatro entradas, que utilice la regla de coincidencia con el prefijo más largo y que reenvíe los paquetes a las interfaces de enlace correctas.

b. Describa cómo determina su tabla de reenvío la interfaz de enlace apropiada para los datagramas con las siguientes direcciones de destino:

11001000 10010001 01010001 01010101
11100001 01000000 11000011 00111100
11100001 10000000 00010001 01110111

P10. Considere una red de datagramas que utiliza direcciones de host de 8 bits. Suponga un router que utiliza las coincidencias con el prefijo más largo y cuya tabla de reenvío es la siguiente:

Coincidencia de prefijo	Interfaz
00	0
010	1
011	2
10	2
11	3

Para cada una de las cuatro interfaces, proporcione el rango asociado de direcciones del host de destino y el número de direcciones contenidas en el rango.

P11. Considere una red de datagramas que utiliza direcciones de host de 8 bits. Suponga un router que utiliza las coincidencias con el prefijo más largo y cuya tabla de reenvío es la siguiente:

Coincidencia de prefijo	Interfaz
1	0
10	1
111	2
en otro caso	3

Para cada una de las cuatro interfaces, proporcione el rango asociado de direcciones del host de destino y el número de direcciones contenidas en el rango.

P12. Sea un router que interconecta tres subredes: Subred 1, Subred 2 y Subred 3. Suponga que se requiere que todas las interfaces de cada una de estas tres subredes tengan el prefijo 223.1.17/24. Suponga también que se requiere que la Subred 1 admita hasta 63 interfaces, la Subred 2 tiene que admitir hasta 95 interfaces y la Subred 3 hasta 16 interfaces. Determine tres direcciones de red (de la forma a.b.c.d/x) que satisfagan estas restricciones.

P13. En la Sección 4.2.2 se ha proporcionado una tabla de reenvío de ejemplo (utilizando la coincidencia con el prefijo más largo). Vuelva a escribir esta tabla de reenvío utilizando la notación a.b.c.d/x en lugar de la notación en binario.

P14. En el Problema P9 se le ha pedido que proporcione una tabla de reenvío (utilizando la regla de coincidencia con el prefijo más largo). Escriba de nuevo esta tabla de reenvío utilizando la notación a.b.c.d/x en lugar de la notación en binario.

P15. Considere un subred cuyo prefijo es 128.119.40.128/26. Proporcione un ejemplo de una dirección IP (de la forma xxx.xxx.xxx.xxx) que pueda ser asignada a esta red. Suponga que un ISP posee el bloque de direcciones 128.119.40.64/25. Suponga que desea crear cuatro subredes a partir de este bloque de direcciones, teniendo cada bloque el mismo número de direcciones IP. ¿Cuáles serán los prefijos (expresados en formato a.b.c.d/x) para las cuatro subredes?

P16. Considere la topología de la Figura 4.17. Denomine a las tres subredes con hosts (comenzando en el sentido horario a partir de las 12:00) como redes A, B y C. Denomine a las subredes que no tienen hosts como redes D, E y F.

a. Asigne direcciones de red a cada una de estas seis subredes, teniendo en cuenta las siguientes restricciones: todas las direcciones tienen que ser asignadas a partir de 214.97.254/23; la subred A tendrá que disponer de las direcciones suficientes como para dar soporte a 250 interfaces; la subred B tendrá que disponer de las direcciones suficientes como para dar soporte a 120 interfaces y la subred C tendrá que disponer de las direcciones suficientes como para dar soporte a 120 interfaces. Por supuesto, las subredes D, E y F deberían poder dar soporte a dos interfaces. Para cada subred, la asignación debería hacerse empleando el formato a.b.c.d/x o a.b.c.d/x – e.f.g.h/y.

b. Utilizando su respuesta al apartado (a), proporcione las tablas de reenvío (utilizando la regla de coincidencia con el prefijo más largo) para cada uno de los tres routers.

P17. Se envía un datagrama de 2.400 bytes por un enlace que tiene una MTU de 700 bytes. Suponga que el datagrama original está marcado con el número de identificación 422.

¿Cuántos fragmentos se generan? ¿Cuáles son los valores de los distintos campos de los datagramas IP generados, relacionados con la fragmentación?

P18. Suponga que el tamaño de los datagramas está limitado a 1.500 bytes (incluyendo la cabecera) entre el host A y el host de destino B. Suponiendo una cabecera IP de 20 bytes, ¿cuántos datagramas se necesitarían para enviar un archivo MP3 de 5 millones de bytes? Explique los cálculos que haya realizado para dar una respuesta.

P19. Considere la red de la Figura 4.22. Suponga que el ISP asigna al router la dirección 24.34.112.235 y que la dirección de la red doméstica es 192.168.1/24.

a. Asigne direcciones a todas las interfaces de la red doméstica.

b. Suponga que cada host tiene dos conexiones TCP activas, todas ellas en el puerto 80 del host 128.119.40.86. Proporcione las seis entradas correspondientes de la tabla de traducciones NAT.

P20. Suponga que está interesado en detectar el número de hosts que hay detrás de un traductor NAT. Observe que la capa IP marca secuencialmente con un número de identificación cada paquete IP. El número de identificación del primer paquete IP generado por un host es un número aleatorio y los números de identificación de los subsiguientes paquetes IP se asignan de forma secuencial. Suponga que todos los paquetes IP generados por los hosts que hay detrás del traductor NAT se envían al exterior.

a. Basándose en esta observación y suponiendo que puede husmear todos los paquetes enviados por NAT al exterior, ¿puede esbozar una técnica sencilla para detectar el número de hosts únicos que hay detrás del traductor NAT? Justifique su respuesta.

b. Si los números de identificación no se asignan secuencialmente sino aleatoriamente, ¿serviría su técnica? Justifique su respuesta.

P21. En este problema se explora el impacto de NAT sobre las aplicaciones P2P. Suponga que un par cuyo nombre de usuario es Arnold descubre mediante una consulta que otro par con el nombre de usuario Bernard tiene un archivo que desea descargar. Suponga también que Bernard y Arnold están detrás de un traductor NAT. Intente deducir una técnica que permita a Arnold establecer una conexión TCP con Bernard sin realizar una configuración NAT específica de la aplicación. Si tiene dificultades para definir una técnica así, explique por qué.

P22. En la Figura 4.27, enumere la rutas de y a u que no contienen ningún bucle.

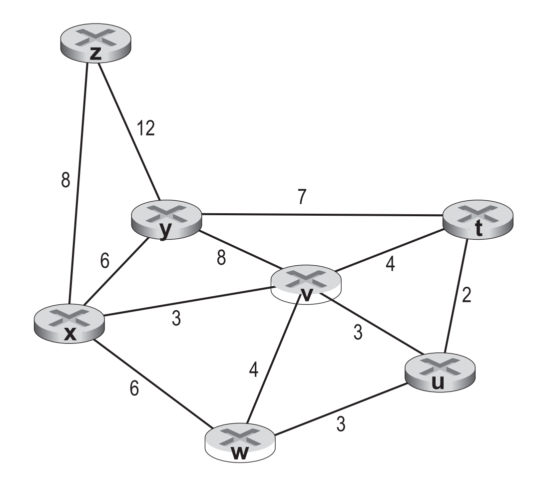
P23. Repita el Problema P22 para las rutas de x a z, z a u y z a w.

P24. Considere la red de la página siguiente. Con los costes de enlace indicados, utilice el algoritmo de la ruta más corta de Dijkstra para calcular la ruta más corta de x a todos los nodos de la red. Muestre cómo funciona el algoritmo calculando una tabla similar a la mostrada en la Tabla 4.3.

P25. Considere la red del Problema P24. Utilizando el algoritmo de Dijkstra y utilizando una tabla similar a la Tabla 4.3, haga lo siguiente:

a. Calcule la ruta más corta desde t a todos los demás nodos de la red.

b. Calcule la ruta más corta desde u a todos los demás nodos de la red.



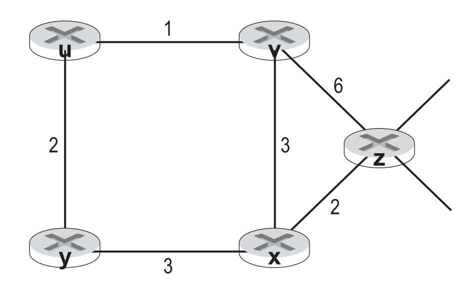
c. Calcule la ruta más corta desde *v* a todos los demás nodos de la red.

d. Calcule la ruta más corta desde *w* a todos los demás nodos de la red.

e. Calcule la ruta más corta desde *y* a todos los demás nodos de la red.

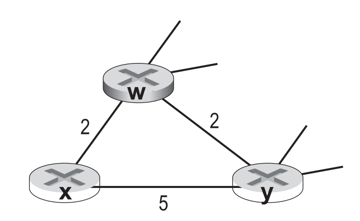
f. Calcule la ruta más corta desde *z* a todos los demás nodos de la red.

P26. Utilice la red que se muestra a continuación y suponga que cada nodo inicialmente conoce los costes hasta cada uno de sus vecinos. Utilizando el algoritmo de vector de distancias, especifique las entradas de la tabla de distancias para el nodo *z*.



P27. Considere una topología general (es decir, no la red concreta mostrada más arriba) y una versión síncrona del algoritmo de vector de distancias. Suponga que en cada iteración un nodo intercambia sus vectores distancia con sus vecinos y recibe los vectores distancia de ellos. Suponiendo que el algoritmo se inicia con cada nodo conociendo sólo los costes de sus vecinos inmediatos, ¿cuál es el número máximo de iteraciones requerido antes de que el algoritmo distribuido converja? Justifique su respuesta.

P28. Considere el fragmento de red mostrado a continuación. *x* sólo tiene dos vecinos conectados, *w* e *y*. *w* tiene una ruta de coste mínimo al destino *u* (no mostrado) de 5 e *y* tiene una ruta de coste mínimo a *u* de 6. Las rutas completas desde *w* e *y* a *u* (y entre *w* e *y*) no se muestran. Todos los costes de enlace de la red tienen valores enteros estrictamente positivos.



a. Indique el vector de distancias de x para los destinos w, y y u.

b. Indique un cambio en el coste del enlace para $c(x,w)$ o $c(x,y)$ tal que x informe a sus vecinos de una nueva ruta de coste mínimo a u, como resultado de ejecutar el algoritmo de vector de distancias.

c. Indique un cambio en el coste del enlace para $c(x,w)$ o $c(x,y)$ tal que x no informe a sus vecinos de una ruta de coste mínimo a u, como resultado de ejecutar el algoritmo de vector de distancias.

P29. Considere la topología de tres nodos mostrada en la Figura 4.30. En lugar de tener los costes de enlace mostrados en dicha figura, los costes de enlace son $c(x,y) = 3$, $c(y,z) = 6$, $c(z,x) = 4$. Calcule las tablas de distancias después del paso de inicialización y después de cada iteración de una versión síncrona del algoritmo de vector de distancias (como hemos hecho anteriormente al explicar la Figura 4.30).

P30. Considere el problema de la cuenta hasta infinito en el enrutamiento por vector de distancias. ¿Se producirá dicho problema si disminuimos el coste de un enlace? ¿Por qué? ¿Qué ocurre si conectamos dos nodos que no tienen un enlace?

P31. Demuestre que al aplicar el algoritmo de vector de distancias en la Figura 4.30, cada valor del vector de distancias $D(x)$ es no creciente y finalmente se estabilizará en un número finito de pasos.

P32. Considere la Figura 4.31. Suponga que existe otro router w, conectado a los routers y y z. Los costes de todos los enlace son los siguientes: $c(x,y) = 4$, $c(x,z) = 50$, $c(y,w) = 1$, $c(z,w)=1$, $c(y,z) = 3$. Suponga que se utiliza inversa envenenada en el algoritmo de enrutamiento por vector de distancias.

a. Cuando el enrutamiento por vector de distancias se estabiliza, los routers w, y y z se informan de sus respectivas distancias a x. ¿Cuáles son los valores de esas distancias?

b. Ahora suponga que el coste del enlace entre x e y aumenta a 60. ¿Se producirá un problema de cuenta hasta infinito aunque se utilice inversa envenenada? ¿Por qué? Si existe el problema de cuenta hasta infinito, entonces ¿cuántas iteraciones serán necesarias para que el enrutamiento por vector de distancias alcance de nuevo un estado estable? Justifique su respuesta.

c. ¿Cómo modificaría $c(y,z)$ para que no existiera el problema de cuenta hasta infinito si $c(y,x)$ cambia de 4 a 60?

P33. Describa cómo puede detectarse en BGP la existencia de bucles en las rutas.

P34. ¿Un router BGP elegirá siempre la ruta sin bucles con la longitud más corta de la secuencia AS-PATH? Justifique su respuesta.

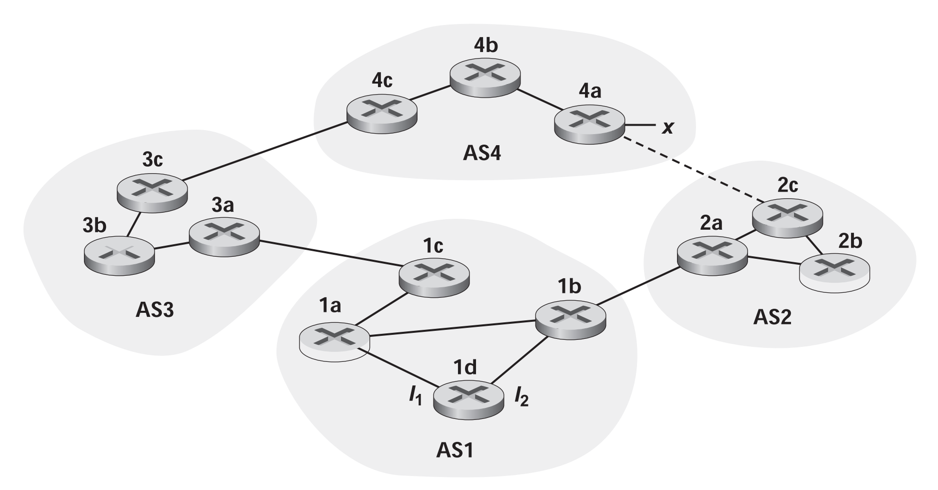
P35. Considere la red mostrada a continuación. Suponga que los sistemas autónomos AS3 y AS2 están ejecutando OSPF como protocolo de enrutamiento interno. Suponga que AS1 y AS4 están ejecutando RIP como protocolo de enrutamiento interno. Suponga por último que se utilizan sesiones eBGP y iBGP para el protocolo de enrutamiento entre sistemas autónomos. Además, inicialmente no existe enlace físico entre AS2 y AS4.

a. ¿De qué protocolo de enrutamiento aprende el router 3c acerca del prefijo *x*: OSPF, RIP, eBGP o iBGP?

b. ¿De qué protocolo de enrutamiento aprende el router 3a acerca de *x*?

c. ¿De qué protocolo de enrutamiento aprende el router 1c acerca de *x*?

d. ¿De qué protocolo de enrutamiento aprende el router 1d acerca de *x*?



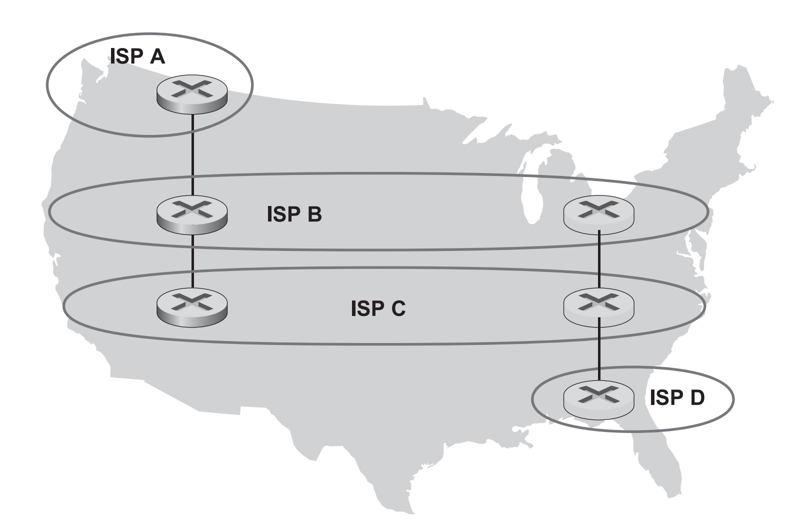
P36. Continuando con el problema anterior, una vez que el router 1d aprende acerca de *x* incluirá una entrada (*x*, *I*) en su tabla de reenvío.

a. Para esta entrada, ¿I será igual a I_1 o a I_2? Explique por qué en una frase.

b. Ahora suponga que existe un enlace físico entre AS2 y AS4 (mostrado mediante una línea de puntos en la figura). Suponga que el router 1d aprende que *x* es accesible a través de AS2 y de AS3. ¿I será igual a I_1 o a I_2? Explique por qué en una frase.

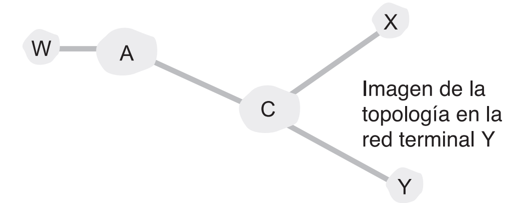
c. Ahora suponga que existe otro sistema autónomo AS5, que conecta la ruta entre AS2 y AS4 (no se muestra en el diagrama). Suponga que el router 1d aprende que *x* es accesible a través de AS2 AS5 AS4, así como de AS3 AS4. ¿I será igual a I_1 o a I_2? Explique por qué en una frase.

P37. Considere la red mostrada en la primera figura de la página siguiente. El ISP B proporciona un servicio troncal nacional al ISP A regional. El ISP C ofrece un servicio troncal nacional al ISP D regional. Cada ISP consta de un sistema autónomo. Los pares B y C se comunican entre sí por dos puntos utilizando BGP. Considere el tráfico que va de A a D. B preferiría manipular dicho tráfico hacia C por el enlace de



la Costa Oeste (de modo que C tendría que absorber el coste de transportar el tráfico a través del país), mientras que C preferiría obtener ese tráfico a través del enlace con B de la Costa Este, en cuyo caso B tendría que transportar el tráfico a través del país. ¿Qué mecanismo BGP podría utilizar C para que B llevara el tráfico de A-a-D al punto de contacto de la Costa Este? Para responder a esta pregunta, tendrá que ahondar en la especificación de BGP.

P38. En la Figura 4.42, considere la información de rutas que llega a las redes terminales (*stub*) W, X y Y. Basándose en la información disponible en W y X, ¿cuáles son sus respectivas imágenes de la topología de la red? Justifique su respuesta. La imagen de la topología en Y se muestra a continuación.



P39. Considere la Figura 4.42. B nunca reenviaría tráfico destinado a Y a través de X basándose en el enrutamiento BGP. Pero existen algunas aplicaciones muy populares en las que los paquetes de datos se dirigen primero a X y luego fluyen hacia Y. Identifique una de tales aplicaciones y describa cómo los paquetes de datos siguen una ruta que no ha sido determinada por el enrutamiento BGP.

P40. En la Figura 4.42, suponga que existe otra red terminal V que es un cliente del ISP A. Suponga que B y C tienen una relación de pares y que A es cliente tanto de B como de C. Suponga también que A preferiría que el tráfico destinado a W procediera sólo de B, y que el tráfico destinado a V procediera de B o de C. ¿Cómo podría anunciar A sus rutas a B y C? ¿Qué rutas del sistema autónomo recibe C?

P41. Considere la red de siete nodos (con los nodos etiquetados de *t* a *z*) del Problema P4. Demuestre que el árbol de coste mínimo con raíz en *z* incluye (como hosts terminales) los nodos *u*, *v*, *w* y *y*. Demuestre informalmente que este árbol es un árbol de coste mínimo.

P42. Considere los dos métodos básicos identificados para llevar a cabo la difusión por emulación de unidifusión y basada en la capa de red (es decir, con ayuda de los routers) y suponga que se utiliza la difusión por árbol de recubrimiento para realizar dicha comunicación en la capa de red. Considere un único emisor y 32 receptores. Suponga que el emisor está conectado a los receptores mediante un árbol binario de routers. Para esta topología, ¿cuál es el coste de enviar un paquete de difusión en los casos de difusión por emulación de unidifusión y basada en la capa de red? Aquí, cada vez que se envía un paquete (o una copia de un paquete) a través de un único enlace se añade una unidad de coste. ¿Qué topología de interconexión entre el emisor, los receptores y los routers hará que se diferencien lo máximo posible el coste de la emulación de unidifusión y el de la verdadera difusión basada en la capa de red? Puede seleccionar tantos routers como desee.

P43. Considere la operación del algoritmo de reenvío de camino inverso (RPF) utilizado en la Figura 4.44. Utilizando la misma topología, determine un conjunto de rutas desde todos los nodos al nodo de origen *A* (e indique estas rutas en un grafo utilizando líneas sombreadas más gruesas como en la Figura 4.44), tal que si esas rutas fueran las rutas de coste mínimo, entonces el nodo *B* recibiría una copia del mensaje de difusión de *A* procedente de los nodos *A*, *C* y *D* bajo RPF.

P44. Considere la topología mostrada en la Figura 4.44. Suponga que todos los enlaces tienen un coste unitario y que el nodo *E* es el origen de las comunicaciones por difusión. Utilizando flechas como las mostradas en la Figura 4.44, indique los enlaces por los que se reenviarán los paquetes utilizando RPF y los enlaces por los que los paquetes no serán reenviados, dado que el nodo *E* es el origen.

P45. Repita el Problema P44 utilizando el grafo del Problema P24. Suponga que *z* es el origen de la difusión y que los costes de los enlaces son los mostrados en el Problema P22.

P46. Considere la topología mostrada en la Figura 4.46 y suponga que cada enlace tiene un coste igual a la unidad. Suponga que se elige el nodo *C* como el nodo central de un algoritmo de enrutamiento por multidifusión basado en un nodo central. Suponiendo que cada router conectado utiliza su ruta de coste mínimo hacia el nodo *C* para enviar mensajes de unión a *C,* dibuje el árbol de enrutamiento basado en un nodo central resultante. ¿Es el árbol resultante el árbol de coste mínimo? Justifique su respuesta.

P47. Repita el Problema P46 utilizando el grafo del Problema P24. Suponga que el nodo central es el nodo *v*.

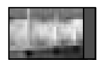
P48. En la Sección 4.5.1 hemos estudiado el algoritmo de enrutamiento de estado de enlaces de Dijkstra para calcular las rutas de unidifusión que son, individualmente, las rutas de coste mínimo desde el origen hacia todos los destinos. Podría pensarse que la unión de estas rutas dará como resultado un **árbol de rutas de unidifusión de coste mínimo** (o un árbol de rutas de unidifusión más cortas, si todos los costes de los enlaces son idénticos). Construyendo un contraejemplo, demuestre que el árbol de la ruta de coste mínimo *no* siempre es igual que el árbol de recubrimiento mínimo.

P49. Considere una red en la que todos los nodos están conectados a otros tres nodos. En un único intervalo de tiempo, un nodo puede recibir todos los paquetes de difusión transmitidos desde sus vecinos, duplicar los paquetes y enviarlos a cada uno de sus vecinos (excepto al nodo que envío un paquete concreto). En el siguiente intervalo de tiempo, los nodos vecinos puede recibir, duplicar y reenviar estos paquetes, y así sucesivamente. Suponga que se utiliza la técnica de inundación no controlada para proporcionar comunicación por difusión en una red así. En el intervalo de tiempo *t*, ¿cuántas copias del paquete de difusión se transmitirán, suponiendo que durante el intervalo de tiempo 1 el nodo de origen transmitió un único paquete de difusión a sus tres vecinos?

P50. Hemos visto en la Sección 4.7 que no existe ningún protocolo de la capa de red que se pueda utilizar para identificar a los hosts que participan en un grupo de multidifusión. Sabiendo esto, ¿cómo pueden aprender las aplicaciones de multidifusión las identidades de los hosts que están participando en un grupo de multidifusión?

P51. Diseñe (proporcione el pseudocódigo) para un protocolo del nivel de aplicación que mantenga las direcciones de host de todos los host participantes en un grupo de multidifusión. Identifique específicamente el servicio de red (unidifusión o multidifusión) que vaya a utilizar su protocolo e indique si su protocolo está enviando mensajes en banda o fuera de banda (con respecto al flujo de datos de aplicación entre los participantes del grupo de multidifusión) y por qué.

P52. ¿Cuál es el tamaño del espacio de direcciones de multidifusión? Suponga ahora que dos grupos de multidifusión seleccionan aleatoriamente una dirección de multidifusión. ¿Cuál es la probabilidad de que elijan la misma dirección? Suponga que 1.000 grupos de multidifusión están activos al mismo tiempo y seleccionan sus direcciones de grupo de multidifusión aleatoriamente. ¿Cuál es la probabilidad de que interfieran entre sí?



Preguntas para la discusión

D1. Localice tres empresas que vendan actualmente routers de alta velocidad. Compare los routers más potentes que comercialicen. ¿Cómo definiría "más potente"?

D2. Utilice el servicio *whois* del Registro americano de números de Internet (http://www.arin.net/whois) para determinar los bloques de direcciones IP de tres universidades. ¿Puede utilizarse el servicio *whois* para determinar con certeza la ubicación geográfica de una dirección IP específica?

D3. ¿Es posible escribir en Java el programa cliente `ping` (utilizando mensajes ICMP)? ¿Por qué?

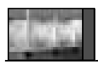
D4. En la Sección 4.4 hemos comentado que la implantación de IPv6 ha sido lenta. ¿Por qué ha sido lenta? ¿Qué se necesita para acelerar su implantación?

D5. Comente algunos de los problemas que NAT crea a la seguridad IPsec (consulte [Phifer 2000]).

D6. Estudie el protocolo UPnP. Describa los mensajes que utiliza un host para reconfigurar un traductor NAT.

D7. Suponga que los sistemas autónomos X y Z no están directamente conectados, pero están conectados a través del sistema autónomo Y. Suponga también que X tiene un acuerdo de comunicación entre pares con Y, y que Y tiene un acuerdo similar con Z. Por último, suponga que Z desea transferir todo el tráfico de Y pero no el de X. ¿Permite BGP implementar esta política a Z?

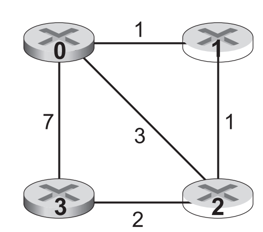
D8. En la Sección 4.7 hemos identificado una serie de aplicaciones de multidifusión. ¿Cuáles de estas aplicaciones están bien adaptadas para el modelo minimalista de servicio de multidifusión de Internet? ¿Por qué? ¿Qué aplicaciones no están especialmente bien adaptadas a este modelo de servicio?



Tareas de programación

En esta tarea de programación, tendrá que escribir un conjunto "distribuido" de procedimientos que implemente un enrutamiento asíncrono distribuido por vector de distancias para la red que se muestra más abajo.

Tendrá que escribir las siguientes rutinas que se "ejecutarán" de forma asíncrona dentro del entorno emulado proporcionado para esta tarea. Para el nodo 0, escribirá las rutinas siguientes:

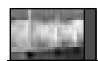


- *rtinit0().* Se llamará a esta rutina una vez al principio de la emulación. *rtinit0()* no tiene argumentos. Debe inicializar su tabla de distancias en el nodo 0 para reflejar los costes directos de 1, 3 y 7 a los nodos 1, 2 y 3, respectivamente. En la figura de encima, todos los enlaces son bidireccionales y los costes en ambas direcciones son idénticos. Después de inicializar la tabla de distancias y cualquier otra estructura de datos que necesiten sus rutinas del nodo 0, deberá entonces enviar a sus vecinos directamente conectados (en este caso, 1, 2 y 3) el coste de sus rutas de coste mínimo a los demás nodos de la red. La información de coste mínimo se envía a los nodos vecinos mediante un paquete de actualización de enrutamiento llamando a la rutina *tolayer2(),* como se describe en la tarea completa. El formato del paquete de actualización de enrutamiento también está descrito en la tarea completa.
- *rtupdate0(struct rtpkt *rcvdpkt)*. Se llamará a esta rutina cuando el nodo 0 reciba un paquete de enrutamiento que le haya enviado uno de sus vecinos directamente conectados. El parámetro **rcvdpkt* es un puntero al paquete que ha recibido. *rtupdate0()* es el "núcleo" del algoritmo de vector de distancias. Los valores recibidos en un paquete de actualización de enrutamiento procedente de algún otro nodo *i* contienen los costes de la ruta más corta actual de *i* hacia todos los demás nodos de la red. *rtupdate0()* utiliza estos valores para actualizar su propia tabla de distancias (como especifica el algoritmo

de vector de distancias). Si su propio coste mínimo a otro nodo cambia como resultado de la actualización, el nodo 0 informa de este cambio del coste mínimo a sus vecinos directamente conectados enviándoles un paquete de enrutamiento. Recuerde que, en el algoritmo de vector de distancias, sólo los nodos directamente conectados intercambiarán paquetes de enrutamiento. Por tanto, los nodos 1 y 2 se comunicarán entre sí, pero los nodos 1 y 3 no lo harán.

Para los nodos 1, 2 y 3 se definen rutinas similares. Por tanto, tendrá que escribir ocho procedimientos en total: *rtinit0(), rtinit1(), rtinit2(), rtinit3(), rtupdate0(), rtupdate1(), rtupdate2()* y *rtupdate3()*. Estas rutinas implementarán conjuntamente un cálculo asíncrono y distribuido de las tablas de distancias para la topología y los costes mostrados en la figura anterior.

Puede encontrar todos los detalles acerca de esta tarea de programación, así como el código C que tendrá que crear en el entorno hardware/software simulado en http://www.awl.com/kurose-ross. También hay disponible una versión Java de la tarea.

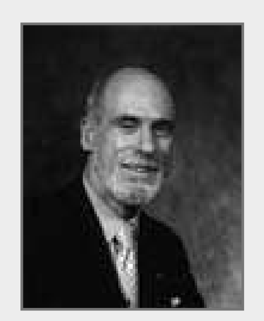


Prácticas de laboratorio con Wireshark

En el sitio web del libro, http://www.awl.com/kurose-ross, encontrará dos prácticas de laboratorio con Wireshark. La primera de ellas examina el funcionamiento del protocolo IP y el formato del datagrama de IP en concreto. La segunda se ocupa del uso del protocolo ICMP en los comandos `ping` y `traceroute`.

ENTREVISTA CON...

Vinton G. Cerf



Vinton G. Cerf es Vicepresidente y jefe de estrategia de Internet de Google. Ha trabajado durante más de 16 años en MCI en distintos puestos, acabando allí su carrera profesional como Vicepresidente senior de Estrategia Tecnológica. Es muy conocido por haber sido co-diseñador de los protocolos TCP/IP y de la arquitectura de Internet. Durante el tiempo que trabajó, entre 1976 y 1982, en la Agencia de Proyectos de Investigación Avanzada de Defensa (DARPA, *Department of Defense Advanced Research Projects Agency*) de Estados Unidos desempeñó un papel crucial dirigiendo el desarrollo de Internet y de las tecnologías de seguridad y de empaquetado de datos relacionadas con Internet. Recibió la Medalla Presidencial de la Libertad en 2005 y la Medalla Nacional de Tecnología en 1997. Es licenciado en Matemáticas por la Universidad de Stanford y Doctor en Ciencias de la Computación por UCLA.

¿Qué le hizo especializarse en el campo de las redes?

Estaba trabajando como programador en UCLA a finales de la década de 1960. Mi trabajo estaba financiado por la Agencia de Proyectos de Investigación Avanzada de Defensa de Estados Unidos (que entonces se llamada ARPA y ahora DARPA). Trabajaba en el laboratorio del profesor Leonard Kleinrock en el Centro de Medidas de Red para la recientemente creada red ARPAnet. El primer nodo de ARPAnet fue instalado en UCLA el 1 de septiembre de 1969. Yo era responsable de programar una computadora que se utilizaba para capturar información de rendimiento acerca de ARPAnet y de devolver esa información para compararla con los modelos matemáticos y las predicciones relativas al rendimiento de la red.

A algunos otros estudiantes y a mi mismo nos encargaron trabajar en los denominados protocolos de nivel de host de ARPAnet, es decir, en los procedimientos y formatos que permitirían que muchos tipos distintos de computadoras interactuaran entre sí a través de la red. Una investigación fascinante en un mundo nuevo (para mí) de la comunicación y la computación distribuida.

Cuando diseñó el protocolo, ¿se imaginaba que IP llegaría a ser tan ubicuo como lo es hoy día?

Cuando Bob Kahn y yo comenzamos a trabajar en el tema en 1973, creo que estábamos muy concentrados en el tema fundamental: cómo hacer que una serie de redes de paquetes heterogéneas interoperaran, partiendo del supuesto de que no podíamos modificar las propias redes. Intentábamos encontrar una forma de poder interconectar un conjunto arbitrario de redes de paquetes de manera transparente, de modo que las computadoras pudieran comunicarse de terminal a terminal sin necesidad de realizar ninguna traducción intermedia. Creo que éramos conscientes de que estábamos trabajando con una tecnología potente y ampliable, pero me parece que no teníamos una imagen clara de lo que podía llegar a ser nuestro mundo con centenares de millones de computadoras interconectadas a través de Internet.

¿Cuál cree que será el futuro de las redes y de Internet? ¿Cuáles cree que son los principales desafíos/obstáculos a los que nos enfrentaremos?

Creo que la propia Internet y las redes en general continuarán evolucionando y expandiéndose. Ya existen suficientes evidencias para afirmar que pronto habrá miles de millones de dispositivos compatibles con Internet, incluyendo equipos tales como teléfonos móviles, frigoríficos, asistentes digitales personales (PDA), servidores domésticos, televisiones, además de la colección habitual de computadoras portátiles, servidores, etc. Entre los principales desafíos se incluyen el soporte para la movilidad, la capacidad de las baterías, la capacidad de los enlaces de acceso a la red y la capacidad de ampliar el núcleo óptico de la red de forma ilimitada. El diseño de una extensión interplanetaria de Internet es un proyecto en el que estoy profundamente involucrado en el Jet Propulsion Laboratory. Asimismo, necesitaremos hacer la transición desde IPv4 [direcciones de 32 bits] a IPv6 [direcciones de 128 bits]. ¡Hay una larga lista de desafíos!

¿Quién le ha servido de inspiración profesionalmente?

Mi colega Bob Kahn; mi director de tesis, Gerald Estrin; mi mejor amigo, Steve Crocker (nos conocimos en la facultad y fue él el que me introdujo en el mundo de las computadoras 1960); y los miles de ingenieros que hacen que Internet continúe evolucionando hoy día.

¿Qué consejo le daría a los estudiantes que inician su andadura en el campo de las redes/Internet?

Que tienen que pensar sin tener en cuenta las limitaciones de los sistemas existentes. Deben tratar de imaginar qué cosas son posibles y a continuación ponerse manos a la obra intentando averiguar cómo ir hasta allí desde el estado actual de las cosas. Es preciso ser capaz de soñar: media docena de colegas y yo hemos estado trabajando en el Jet Propulsion Laboratory en el diseño de una extensión interplanetaria de la Internet terrestre. Puede que se tarden décadas en implementar esto, misión a misión, pero parafraseando un conocido dicho: “El hombre debe poder llegar más allá de donde su mano alcanza; ¿o para qué son los cielos si no?”.



CAPÍTULO 5

La capa de enlace y las redes de área local

En el capítulo anterior hemos visto que la capa de red proporciona un servicio de comunicaciones entre dos hosts. Como se muestra en la Figura 5.1, esta ruta de comunicación está compuesta por una serie de enlaces de comunicaciones, que comienzan en el host de origen, pasan a través de una serie de routers y terminan en el host de destino. A medida que continuamos bajando por la pila de protocolos, desde la capa de red a la capa de enlace, surge de forma natural la pregunta de cómo se envían los paquetes a través de los *enlaces individuales* que forman la ruta de comunicación de terminal a terminal. ¿Cómo se encapsulan los datagramas de la capa de red en las tramas de la capa de enlace para la transmisión a través de un enlace individual? ¿Pueden proporcionar los protocolos de la capa de enlace una transferencia de datos fiable router a router? ¿Pueden utilizarse diferentes protocolos de la capa de enlace en los distintos enlaces que forman la ruta de comunicaciones? Responderemos a éstas y otras importantes cuestiones a lo largo del capítulo.

Al analizar la capa de enlace, nos encontramos con que hay dos tipos de canales fundamentalmente distintos de la capa de enlace. El primer tipo está compuesto por los canales de difusión, que son comunes en las redes de área local (LAN), en las redes LAN inalámbricas, en las redes por satélite y en las redes de acceso híbridas de fibra y cable coaxial (HFC). En un canal de difusión hay muchos hosts conectados a un mismo canal de comunicaciones, por lo que se hace necesario utilizar lo que se denomina un protocolo de acceso al medio para coordinar las transmisiones y evitar que las tramas transmitidas colisionen. El segundo tipo de canal de la capa de enlace es el canal de comunicaciones punto a punto, como el que existe entre dos routers o entre un módem de acceso telefónico residencial y el router de un ISP. La coordinación del acceso a un enlace punto a punto es trivial, pero sigue habiendo problemas importantes relativos al entramado, a la transferencia de datos fiable, a la detección de errores y al control de flujo.

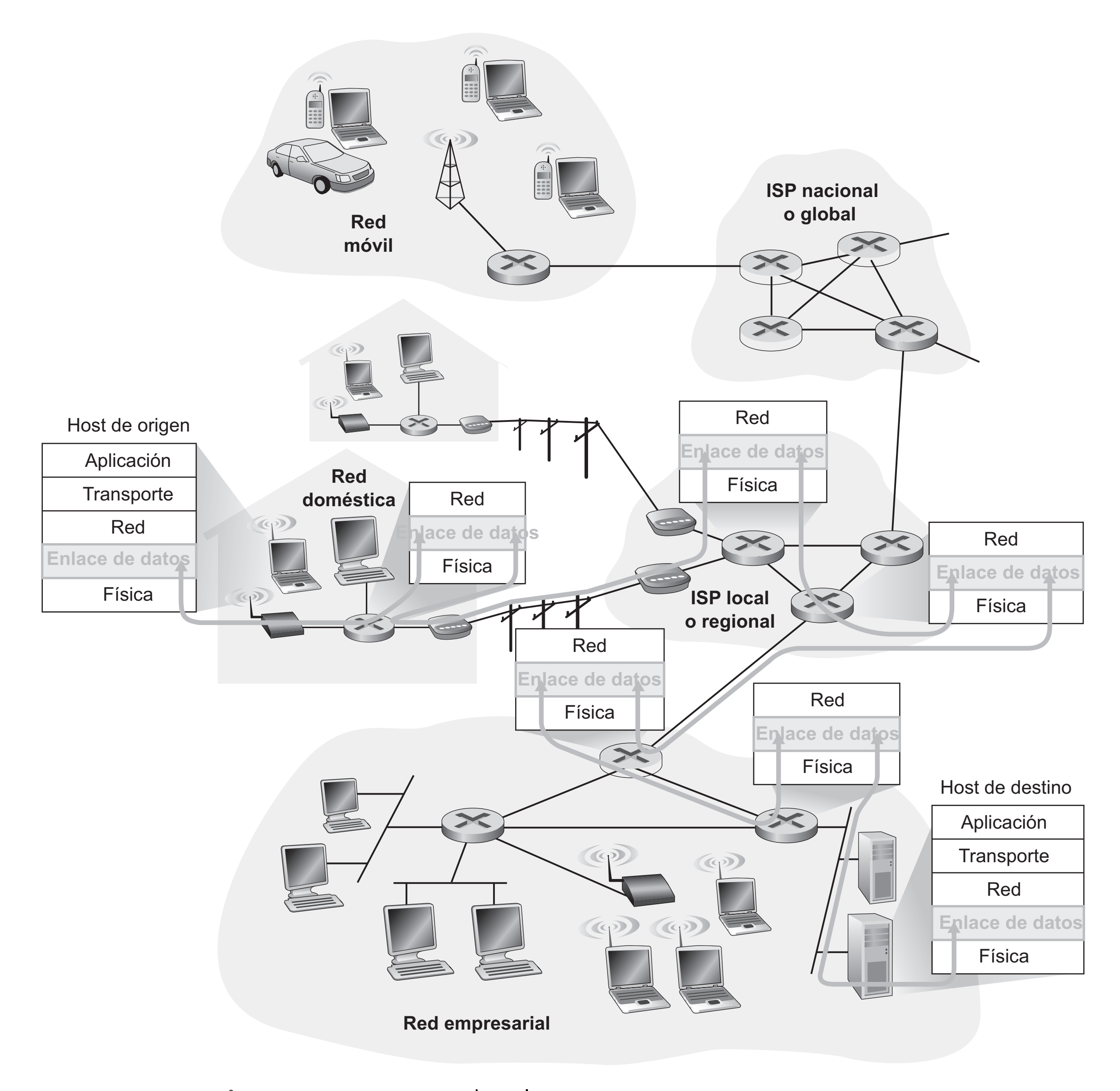


Figura 5.1 • La capa de enlace.

En este capítulo exploraremos diversas tecnologías importantes de la capa de enlace. Examinaremos en profundidad Ethernet, que es con mucho la tecnología de LAN cableada más prevalente. También examinaremos el Protocolo punto a punto (PPP, *Point-to-Point Protocol*), que es el protocolo preferido para los hosts residenciales con conexión mediante acceso telefónico. Aunque WiFi, y más en general las redes LAN inalámbricas, es un tema relacionado con la capa de enlace, pospondremos nuestro estudio de este importante tema hasta el Capítulo 6, que está dedicado a las redes de computadoras inalámbricas y a la movilidad.

5.1 Capa de enlace: introducción y servicios

Comencemos proporcionando alguna terminología útil. En este capítulo nos resultará conveniente referirnos a los hosts y los routers simplemente como **nodos** ya que, como pronto veremos, no nos va a preocupar especialmente si un determinado nodo es un router o un host. También nos referimos a los canales de comunicación que conectan nodos adyacentes a lo largo de la ruta de comunicaciones con el nombre de **enlaces**. Para que un datagrama pueda ser transferido desde el host de origen al de destino, debe moverse a través de cada uno de los *enlaces individuales* que forman la ruta terminal a terminal. En un determinado enlace, un nodo transmisor encapsula el datagrama en una trama de la capa de enlace y transmite la trama a través del enlace; el nodo receptor recibe entonces la trama y extrae el datagrama.

5.1.1 Servicios proporcionados por la capa de enlace

Para transmitir un datagrama a través de un enlace individual se utiliza un protocolo de la capa de enlace. El **protocolo de la capa de enlace** define el formato de los paquetes intercambiados por los nodos situados en los extremos del enlace, así como las acciones que estos nodos llevan a cabo cuando se envían y reciben los paquetes. Recuerde del Capítulo 1 que las unidades de datos intercambiadas por un protocolo de la capa de enlace se denominan **tramas (*frames*)**, y que cada trama de la capa de enlace suele encapsular un datagrama de la capa de red. Como pronto veremos, las acciones que el protocolo de la capa de enlace lleva a cabo a la hora de enviar y transmitir tramas incluye la detección de errores, la retransmisión, el control de flujo y el acceso aleatorio. Como ejemplos de protocolos de la capa de enlace podemos citar Ethernet, las redes LAN inalámbricas 802.11 (también denominadas WiFi), token ring y PPP. Hablaremos en detalle de muchos de estos protocolos en la segunda mitad de este capítulo.

Mientras que la capa de red tiene asignada la tarea de mover los segmentos de la capa de transporte terminal a terminal desde el host de origen al host de destino, el protocolo de la capa de enlace tiene la tarea nodo a nodo, algo más simple, de mover los datagramas de la capa de red a través de un *único enlace* dentro de la ruta. Una característica importante de la capa de enlace es que un mismo datagrama puede ser transportado por diferentes protocolos de la capa de enlace en los distintos enlaces que forman la ruta. Por ejemplo, un datagrama podría ser transportado mediante Ethernet en el primer enlace, mediante PPP en el último enlace y mediante un protocolo WAN de la capa de enlace en los enlaces intermedios. Es importante observar que los servicios proporcionados por los diferentes protocolos de la capa de enlace a lo largo de una ruta terminal a terminal pueden ser distintos. Por ejemplo, algunos protocolos de la capa de enlace proporcionan una entrega fiable, mientras que otros no lo hacen. Por tanto, la capa de red debe ser capaz de realizar su trabajo terminal a terminal en presencia de un conjunto heterogéneo de servicios individuales de la capa de enlace.

Para poder comprender el funcionamiento de la capa de enlace y cómo ésta se relaciona con la capa de red, vamos a utilizar una analogía del sector del transporte. Piense en una agencia de viajes que planifica un viaje para un turista que quiere ir desde Princeton, Nueva Jersey, a Lausanne, Suiza. La agencia de viajes decide que lo más conveniente para el turista es tomar una limusina en Princeton hasta el aeropuerto JFK, luego un avión desde dicho aeropuerto hasta el aeropuerto de Ginebra y, finalmente, un tren desde el aeropuerto de Ginebra

hasta la estación de tren de Lausanne. Una vez que la agencia de viajes hace las tres reservas es responsabilidad de la empresa de limousines de Princeton llevar al turista hasta el aeropuerto JFK; de la misma forma, será responsabilidad de la compañía aérea transportar al turista desde el aeropuerto JFK al de Ginebra y será responsabilidad de la compañía ferroviaria Suiza llevar al turista desde Ginebra hasta Lausanne. Cada uno de estos tres segmentos del viaje es un segmento "directo" entre dos ubicaciones "adyacentes". Observe que los tres segmentos de transporte son gestionados por diferentes empresas y utilizan modos de transporte completamente distintos (limousine, avión y tren). Aunque los modos de transporte son diferentes, todos ellos proporcionan el servicio básico de transportar pasajeros desde una ubicación a otra adyacente. En esta analogía del sector del transporte, el turista sería un datagrama, cada uno de los segmentos de transporte sería un enlace de comunicaciones, el modo de transporte sería un protocolo de la capa de enlace y la agencia de viajes sería un protocolo de enrutamiento.

Aunque el servicio básico de cualquier capa de enlace es mover un datagrama desde un nodo hasta otro adyacente a través de un único enlace de comunicaciones, los detalles del servicio proporcionado pueden variar de un protocolo de la capa de enlace a otro. Entre los posibles servicios que un protocolo de la capa de enlace puede ofrecer se incluyen:

- *Entramado.* Casi todos los protocolos de la capa de enlace encapsulan cada datagrama de la capa de red dentro de una trama de la capa de enlace antes de transmitirla a través del enlace. Una trama consta de un campo de datos, en el que se inserta el datagrama de la capa de red, y de una serie de campos de cabecera. (Una trama también puede incluir campos de cola; sin embargo, utilizaremos el término campos de cabecera para referirnos tanto a los de cabecera como a los de cola.) La estructura de la trama está especificada por el protocolo de la capa de enlace. Veremos diferentes formatos de trama cuando examinemos una serie de protocolos específicos de la capa de enlace en la segunda mitad de este capítulo.
- *Acceso al enlace.* Un protocolo de control de acceso al medio (MAC, *Medium Access Control*) especifica las reglas que se utilizan para transmitir una trama a través del enlace. Para los enlaces punto a punto que tengan un único emisor en un extremo del enlace y un único receptor en el otro extremo, el protocolo MAC es muy simple (o no existe): el emisor puede enviar una trama siempre que el enlace esté inactivo. El caso más interesante es cuando hay varios nodos compartiendo un mismo enlace de difusión, en cuyo caso se presenta el denominado problema del acceso múltiple. En ese caso, el protocolo MAC sirve para coordinar la transmisión de las tramas de los múltiples nodos; estudiaremos los protocolos MAC en detalle en la Sección 5.3.
- *Entrega fiable.* Cuando un protocolo de la capa de enlace proporciona un servicio de entrega fiable, garantiza que va a transportar cada datagrama de la capa de red a través del enlace sin que se produzcan errores. Recuerde que ciertos protocolos de la capa de transporte (como TCP) también proporcionan un servicio de entrega fiable. De forma similar a los servicios de entrega fiable de la capa de transporte, el servicio de entrega fiable de la capa de enlace suele implementarse mediante reconocimientos y retransmisiones (véase la Sección 3.4). A menudo se utiliza un servicio de entrega fiable de la capa de enlace en aquellos enlaces que suelen presentar altas tasas de error, como por ejemplo en los enlaces inalámbricos, con el objetivo de corregir los errores localmente (en el enlace en el que se producen los errores), en lugar de obligar a que un protocolo de la capa de transporte o de la de aplicación realice una retransmisión de datos terminal a ter-

minal. Sin embargo, la entrega fiable en la capa de enlace puede considerarse una sobrecarga innecesaria en aquellos enlaces que tengan una baja tasa de errores de bit, incluyendo los enlaces de fibra, los coaxiales y muchos enlaces de cobre de par trenzado. Por esta razón, muchos protocolos de la capa de enlace para enlaces cableados no proporcionan un servicio de entrega fiable.

- *Control de flujo.* Los nodos situados en cada extremo de un enlace tienen una capacidad limitada de almacenamiento en buffer de las tramas. Esto puede ser un problema cuando el nodo receptor puede recibir las tramas a más velocidad de la que puede procesarlas. Sin un control de flujo, el buffer del receptor puede desbordarse con lo que las tramas se perderían. De forma similar a lo que sucede en la capa de transporte, el protocolo de la capa de enlace puede proporcionar un mecanismo de control de flujo para evitar que el nodo emisor al otro lado del enlace abrume al nodo receptor situado en el otro extremo.
- *Detección de errores.* El hardware de la capa de enlace en un nodo receptor pudiera llegar a decidir, incorrectamente, que un bit contenido en una trama es cero cuando fue transmitido como un uno, y viceversa. Dichos errores de bit se introducen debido a la atenuación de las señales y al ruido electromagnético. Puesto que no existe ninguna necesidad de reenviar un datagrama que contenga un error, muchos protocolos de la capa de enlace proporcionan un mecanismo para detectar dichos errores de bit. Esto se lleva a cabo haciendo que el nodo transmisor incluya bits de detección de errores en la trama y que el nodo receptor realice una comprobación de errores. Recuerde de los Capítulos 3 y 4 que las capas de transporte y de red de Internet también ofrecen una forma limitada de detección de errores: la suma de comprobación de Internet. La detección de errores en la capa de enlace normalmente es más sofisticada y se implementa en hardware.
- *Corrección de errores.* La corrección de errores es similar a la detección de errores, salvo porque el receptor no sólo detecta si hay bits erróneos en la trama, sino que también determina exactamente en qué puntos de la trama se han producido los errores (y luego corrige esos errores). Algunos protocolos proporcionan corrección de errores en la capa de enlace sólo para la cabecera del paquete en lugar de para el paquete completo. Hablaremos de la detección y corrección de errores en la Sección 5.2.
- *Semiduplex y full-duplex.* Con la transmisión full-duplex, los nodos de ambos extremos de un enlace pueden transmitir paquetes al mismo tiempo. Sin embargo, con la transmisión semiduplex un mismo nodo no puede transmitir y recibir al mismo tiempo.

Como hemos indicado anteriormente, muchos de los servicios proporcionados por la capa de enlace presentan notables paralelismos con los servicios proporcionados en la capa de transporte. Por ejemplo, tanto la capa de enlace como la capa de transporte pueden proporcionar un servicio de entrega fiable. Aunque los mecanismos utilizados para garantizar una entrega fiable en las dos capas son similares (véase la Sección 3.4), los dos servicios de entrega fiable no son idénticos. Un protocolo de transporte proporciona una entrega fiable de segmentos entre dos procesos, en modo terminal a terminal; un protocolo de la capa de enlace fiable proporciona una entrega fiable de tramas entre dos nodos conectados por un único enlace. De forma similar, los protocolos de la capa de transporte como los de la capa de enlace pueden proporcionar servicios de control de flujo y de detección de errores, pero de nuevo el control de flujo en un protocolo de la capa de transporte se proporciona en modo terminal a terminal, mientras que en un protocolo de la capa de enlace se proporciona entre dos nodos adyacentes.

5.1.2 ¿Dónde se implementa la capa de enlace?

Antes de profundizar en los detalles de la capa de enlace, consideremos la cuestión de *dónde* se implementa esta capa. Nos centraremos aquí en un sistema terminal, puesto que ya hemos visto en el Capítulo 4 cómo se implementa la capa de enlace en una tarjeta de línea de un router. ¿Cómo se implementa la capa de enlace de un host, por hardware o por software? ¿Se implementa en una tarjeta o chip separados? ¿Cómo se realiza la interfaz con el resto del hardware del host y con el resto de los componentes del sistema operativo?

La Figura 5.2 muestra la arquitectura típica de un host. En su mayor parte, la capa de enlace se implementa en un **adaptador de red**, también denominado a veces **Tarjeta de interfaz de red (NIC, *Network Interface Card*)**. El corazón de la tarjeta adaptador de red es el controlador de la capa de enlace, que normalmente es un único chip de propósito especial que implementa muchos de los servicios de la capa de enlace (entramado, acceso al enlace, control de flujo, detección de errores, etc.) identificados en la sección anterior. Por tanto, buena parte de la funcionalidad del controlador de la capa de enlace se implementa por hardware. Por ejemplo, el controlador 8254*x* de Intel [Intel 2009] implementa los protocolos Ethernet que estudiaremos en la Sección 5.5; el controlador Atheros AR5006 [Atheros 2009] implementa los protocolos WiFi 802.11 que estudiaremos en la Sección 6.3. Hasta finales de la década de 1990, la mayoría de los adaptadores de red eran tarjetas físicamente separadas, como por ejemplo las tarjetas PCMCIA o una tarjeta insertable que podía introducirse en una ranura de tarjeta PCI del PC, pero los adaptadores de red se suelen integrar cada vez más en la placa base del host, utilizando una configuración que se denomina "LAN sobre placa base".

En el lado emisor, el controlador toma un datagrama que haya sido creado y almacenado en la memoria del host por las capas superiores de la pila de protocolos, encapsula ese

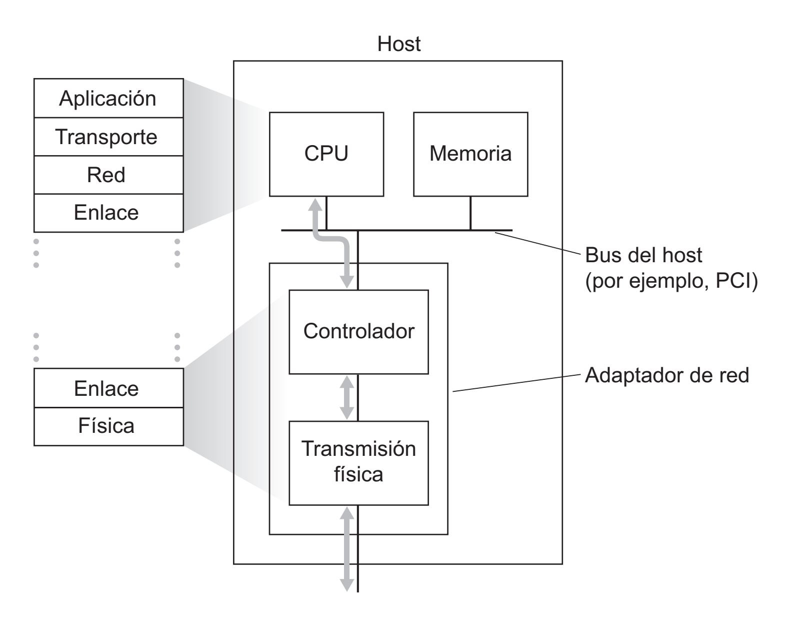


Figura 5.2 • Adaptador de red: su relación con otros componentes del host y con la funcionalidad de la pila de protocolos.

datagrama en una trama de la capa de enlace (rellenando los diversos campos de la trama) y luego transmite la trama al enlace de comunicaciones, de acuerdo con el protocolo de acceso al enlace. En el lado receptor, un controlador recibe la trama completa y extrae el datagrama de la capa de red. Si la capa de enlace realiza detección de errores, entonces será el controlador del emisor quien se encargue de configurar los bits de detección de errores en la cabecera de la trama, mientras que el controlador del receptor llevará a cabo la detección de errores. Si la capa de enlace realiza control de flujo, entonces los controladores del emisor y del receptor intercambian información de control de flujo de modo que el emisor envíe las tramas a una velocidad que el receptor sea capaz de aceptar.

La Figura 5.2 muestra un adaptador de red conectado a un bus del host (por ejemplo, un bus PCI o PCI-X), de modo que a ojos de los restantes componentes del host se parece bastante a cualquier otro dispositivo de E/S. La Figura 5.2 también muestra que, mientras que la mayor parte de la capa de enlace está implementada en el hardware de la tarjeta de interfaz, una parte de esa capa se implementa en un software que se ejecuta en la CPU del host. Los componentes software de la capa de enlace, normalmente implementan la función de más alto nivel de esa capa, como por ejemplo la recepción del datagrama desde la capa de red, el ensamblado de la información de direccionamiento de la capa de enlace y la activación del hardware del controlador. En el lado receptor, el software de la capa de enlace responde a las interrupciones procedentes del controlador (por ejemplo, debidas a la recepción de una o más tramas), se encarga de gestionar las condiciones de error y pasa el datagrama hacia la capa de red. Por tanto, la capa de enlace es una combinación de hardware y software; *es el lugar* de la pila de protocolos en donde el software se encuentra con el hardware. En [Intel 2009] puede encontrar una introducción comprensible (así como una descripción detallada) del controlador 8254*x*, desde el punto de vista del programador de software.

La Figura 5.3 muestra las tarjetas adaptadoras de emisión y recepción. Dado que la funcionalidad principal del protocolo de la capa de enlace está implementada en el controlador, los adaptadores son unidades semi-autónomas cuyo trabajo consiste en transferir una trama de un adaptador a otro. Hay diversos investigadores que han analizado la posibilidad de integrar más funcionalidad (además del procesamiento de la capa de enlace) en los adaptadores de red. El controlador 8254*x*, por ejemplo, puede calcular la suma de comprobación TCP/UDP y la suma de comprobación de la cabecera IP por hardware, lo que significa que hay funcionalidad de las capas de red y de transporte que está implementada en el controlador de la capa de enlace. Aunque esto pueda parecer una sorprendente violación del principio de división en capas, la ventaja es que las sumas de comprobación pueden calcularse

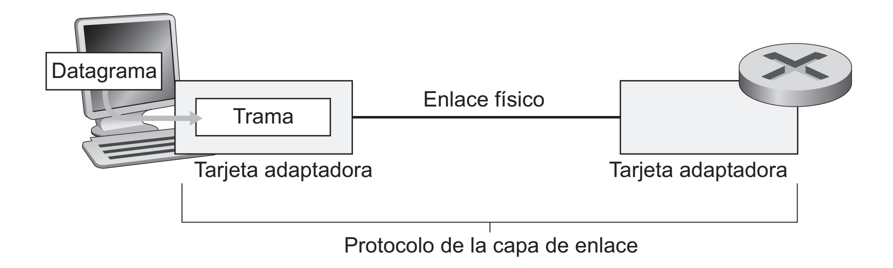


Figura 5.3 • Adaptadores de red comunicándose: un datagrama de la capa de red encapsulado en una trama de la capa de enlace.

mucho más rápidamente mediante hardware que mediante software, tanto que uno se sentiría tentado de ignorar esta violación de las normas. [Mogul 2003] proporciona un interesante análisis de los pros y contras de llevar a cabo procesamiento TCP en un adaptador. [Kim 2005] investiga la conveniencia de llevar a cabo funcionalidades de capas más altas (caché HTTP) en el adaptador.

5.2 Técnicas de detección y corrección de errores

En la sección anterior hemos indicado que la **detección y corrección de errores de nivel bit** (la detección y corrección de los bits corruptos en una trama de la capa de enlace enviada desde un nodo a otro nodo vecino físicamente conectado a él) son dos servicios ofrecidos a menudo por la capa de enlace. Hemos visto en el Capítulo 3 que en la capa de transporte se ofrecen también a menudo los servicios de detección y corrección de errores. En esta sección examinaremos algunas de las técnicas más simples que pueden utilizarse para detectar y, en algunos casos, corregir dichos errores de bit. Un tratamiento completo de la teoría e implementación de estas técnicas constituye por sí mismo el tema de muchos libros de texto (por ejemplo, [Schwartz 1980] o [Bertsekas 1991]), pero el tratamiento que realizaremos aquí es necesariamente breve. Nuestro objetivo es desarrollar simplemente en el lector una percepción intuitiva de las capacidades proporcionadas por las técnicas de detección y corrección de errores, y ver cómo funcionan unas cuantas técnicas simples y cómo se emplean en la práctica dentro de la capa de enlace.

La Figura 5.4 ilustra la configuración para nuestro análisis. En el nodo emisor, los datos que hay que proteger frente a los errores de bit, *D,* se complementan con una serie de bits de detección y corrección de errores (*EDC, Error Detection Correction*). Normalmente, los datos que hay que proteger incluyen no sólo el datagrama recibido de la capa de red para su transmisión a través del enlace, sino también la información de direccionamiento de la capa de enlace, los números de secuencia y otros campos de la cabecera de la trama de enlace. Tanto *D* como *EDC* se envían hacia el nodo receptor en una trama de la capa de enlace. En el nodo receptor, se recibe una secuencia de bits formada por *D* y *EDC* . Observe que *D* y *EDC* pueden diferir de la secuencias *D* y *EDC* originales, como resultado de alteraciones de los bits sufridas durante el tránsito.

El desafío para el receptor consiste en determinar si *D* coincide con la secuencia *D* original, teniendo en cuenta que lo que único que ha recibido son las secuencias *D* y *EDC* . Es importante observar las palabras exactas utilizadas en el símbolo de decisión mostrado en la Figura 5.4 (lo que preguntamos es si se ha detectado un error, no si se ha producido un error). Las técnicas de detección y corrección de errores permiten al receptor detectar en ocasiones, *pero no siempre,* que se han producido errores en los bits. Incluso utilizando bits de detección de errores pueden seguir existiendo **errores de bit no detectados**; es decir, el receptor podría perfectamente ser inconsciente de que la información recibida contiene errores en los bits. Como consecuencia, el receptor podría entregar un datagrama corrupto a la capa de red, o no ser consciente de que el contenido de un campo de la cabecera de la trama se ha corrompido. Por tanto, lo que intentaremos será elegir un esquema de detección de errores que haga que la probabilidad de que se produzcan estos casos sea pequeña. Por regla general, cuanto más sofisticadas son las técnicas de detección y corrección de errores (es decir, cuanto menor sea la probabilidad de que se produzcan errores de bit que no sean

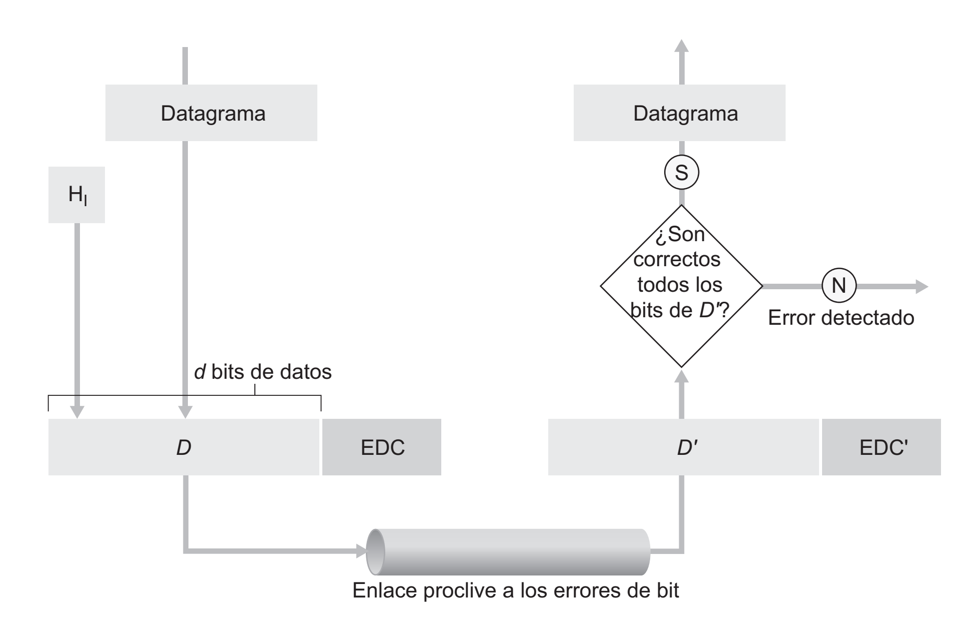


Figura 5.4 • Escenario para la detección y corrección de errores.

detectados), mayores son los recursos adicionales necesarios: harán falta más cálculos para obtener y transmitir un mayor número de bits de detección y corrección de errores.

Examinemos ahora tres técnicas para detectar los errores en los datos transmitidos: comprobaciones de paridad (para ilustrar las ideas básicas que subyacen a las técnicas de detección y corrección de errores), métodos basados en suma de comprobación (que suelen utilizarse más en la capa de transporte) y códigos de redundancia cíclica (que normalmente se emplean más en el adaptador encargado en la capa de enlace).

5.2.1 Comprobaciones de paridad

Quizá la forma más simple de detección de errores sea el uso de un único **bit de paridad**. Suponga que la información que hay que enviar, D en la Figura 5.5, tiene d bits. En un esquema de paridad par, el emisor simplemente incluye un bit adicional y selecciona su valor de modo que el número total de 1s en los $d + 1$ bits (la información original más un bit de paridad) sea par. En los esquemas de paridad impar, el valor del bit de paridad se selecciona de modo que exista un número impar de 1s. La Figura 5.5 ilustra un esquema de paridad par, en el que el único bit de paridad se almacena en un campo separado.

La operación del receptor también es muy simple cuando se utiliza un único bit de paridad. El receptor sólo necesita contar el número de 1s dentro de los $d + 1$ bits recibidos. Si se

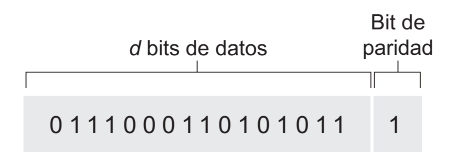


Figura 5.5 • Paridad par de un bit.

está utilizando un esquema de paridad par y se encuentra un número impar de bits con valor 1, el receptor sabrá que se ha producido al menos un error de bit. De forma más precisa, sabrá que se ha producido un número *impar* de errores de bit.

Pero, ¿qué sucede si se produce un número par de errores de bit? Puede comprobar fácilmente que estoy haría que el error no fuera detectado. Si la probabilidad de que se produzcan errores en los bits es pequeña y si podemos asumir que los errores que se producen en dos bits sucesivos tienen lugar de forma independiente, la probabilidad de que se produzcan múltiples errores de bit en un único paquete será extremadamente pequeña. En estos casos, podría bastar con utilizar un único bit de paridad. Sin embargo, las medidas realizadas muestran que los errores más que tener lugar independientemente, suelen agruparse formando "ráfagas". En condiciones de ráfagas de error, la probabilidad de que se produzcan errores no detectados en una trama protegida mediante un único bit de paridad puede aproximarse al 50 por ciento [Spragins 1991]. Evidentemente, necesitamos un esquema de detección de errores más robusto (y, afortunadamente, en la práctica suelen emplearse esos otros esquemas). Pero antes de examinar los esquemas de detección de errores utilizados en la práctica, vamos a analizar una generalización simple del mecanismo de un único bit de paridad; esta generalización nos permitirá comprender mejor las técnicas de corrección de errores.

La Figura 5.6 muestra una generalización bidimensional del esquema basado en un único bit de paridad. Aquí, los d bits de D se dividen en i filas y j columnas. Para cada una de esas filas y columnas calculamos un valor de paridad. Los $i + j + 1$ bits de paridad resultantes serán los bits de detección de errores utilizados en la trama de la capa de enlace.

Suponga ahora que se produce un único error de bit en los d bits de información originales. Con este esquema de **paridad bidimensional**, detectaremos el error en la paridad tanto de la columna como de la fila que contienen el bit erróneo. De este modo, el receptor no sólo podrá *detectar* el hecho de que se ha producido un único error de bit, sino que puede utilizar los índices de la columna y de la fila que presentan errores de paridad para identificar realmente el bit corrompido y *corregir* dicho error. La Figura 5.6 muestra un ejemplo en el que el bit de valor 1 en la posición (2, 2) está corrompido, habiéndose cambiado por un 0; se trata de un error que es tanto detectable como corregible en el receptor. Aunque nuestra explicación se ha centrado en los d bits originales de información, un único error dentro de los propios bits de paridad también es detectable y corregible. Los esquemas de paridad bidimensional también pueden detectar (¡pero no corregir!) cualquier combinación de dos errores dentro de un paquete. En los problemas incluidos al final del capítulo se analizan otras propiedades del esquema de paridad bidimensional.

La capacidad del receptor para detectar y corregir errores a la vez se conoce con el nombre de **Corrección de errores hacia adelante (FEC, Forward Error Correction)**. Estas técnicas se suelen utilizar comúnmente en los dispositivos de almacenamiento y reproducción de audio, como por ejemplo los CD de audio. En un entorno de red, las técnicas FEC también pueden emplearse por sí mismas o en conjunción con técnicas ARQ de la capa de enlace, similares a las que hemos examinado en el Capítulo 3. Las técnicas FEC son valiosas porque pueden reducir el número de retransmisiones realizadas por el emisor pero, lo que quizá sea más importante, también permiten la corrección inmediata de errores en el receptor, lo que evita tener que esperar el retardo de propagación de ida y vuelta necesario para que el emisor reciba un paquete NAK y para que el paquete retransmitido se propague de nuevo hacia el receptor; ésta es una característica potencialmente muy importante para las aplicaciones de red en tiempo real [Rubenstein 1998] o para los enlaces (como los enla-

Paridad de fila

Paridad de columna

$d_{1,1}$	. . .	$d_{1,j}$	$d_{1,j+1}$
$d_{2,1}$	. . .	$d_{2,j}$	$d_{2,j+1}$
. . .	. . .	. . .	. . .
$d_{i,1}$	. . .	$d_{i,j}$	$d_{i,j+1}$
$d_{i+1,1}$	. . .	$d_{i+1,j}$	$d_{i+1,j+1}$

Sin errores

1	0	1	0	1	1
1	1	1	1	0	0
0	1	1	1	0	1
0	0	1	0	1	0

Error corregible de un único bit

1	0	1	0	1	1
1	0	1	1	0	0
0	1	1	1	0	1
0	0	1	0	1	0

Error de paridad

Error de paridad

Figura 5.6 • Paridad par bidimensional.

ces en el espacio profundo) que presenten grandes retardos de propagación. Entre las referencias en las que se examina el uso de las técnicas FEC en los protocolos de control de errores podemos citar [Biersack 1992; Nonnenmacher 1998; Byers 1998; Shacham 1990].

5.2.2 Métodos basados en suma de comprobación

En las técnicas de suma de comprobación, los *d* bits de datos de la Figura 5.5 se tratan como una secuencia de enteros de *k* bits. Un método simple de suma de comprobación consiste en sumar estos enteros de *k* bits y utilizar la suma resultante como bits de detección de errores. La **suma de comprobación de Internet** está basada en este enfoque: los bytes de datos se tratan como enteros de 16 bits y se suman. Entonces, se utiliza el complemento a 1 de esta suma para formar la suma de comprobación de Internet que se incluye en la cabecera del segmento. Como se ha visto en la Sección 3.3, el receptor comprueba la suma de comprobación calculando el complemento a 1 de la suma de los datos recibidos (incluyendo la suma de comprobación) y comprobando si el resultado tiene todos los bits a 1. Si alguno de los bits es un 0, eso indicará que se ha producido un error. El documento RFC 1071 explica en detalle el algoritmo de suma de comprobación de Internet y su implementación. En los protocolos TCP y UDP, la suma de comprobación de Internet se calcula sobre todos los campos (incluyendo los campos de cabecera y de datos). En IP, la suma de comprobación se calcula sobre la cabecera IP (dado que el segmento UDP o TCP ya tiene su propia suma de comprobación). En otros protocolos, como por ejemplo XTP [Strayer 1992], se calcula una suma de comprobación sobre la cabecera y otra sobre todo el paquete.

Los métodos de suma de comprobación requieren relativamente poca sobrecarga de paquete. Por ejemplo, la suma de comprobación en TCP y UDP sólo utiliza 16 bits. Sin embargo, estas sumas proporcionan una protección relativamente débil frente a los errores si las comparamos con las comprobaciones de redundancia cíclica, de las que hablaremos a continuación y que se utilizan a menudo en la capa de enlace. Una pregunta bastante natural llegados a este punto sería: ¿por qué se utilizan sumas de comprobación en la capa de transporte y comprobaciones de redundancia cíclica en la capa de enlace? Recuerde que normalmente la capa de transporte se implementa por software en un host como parte del sistema operativo del mismo. Puesto que el mecanismo de detección de errores de la capa de transporte se implementa por software, es importante utilizar un esquema de detección de errores simple y rápido, como por ejemplo el de la sumas de comprobación. Por el contrario, la detección de errores en la capa de enlace se implementa en un hardware dedicado dentro de las tarjetas adaptadoras, pudiendo dicho hardware realizar rápidamente las operaciones CRC más complejas. Feldmeier [Feldmeier 1995] presenta una serie de técnicas de implementación rápida en software no sólo para códigos de suma de comprobación ponderados, sino también para códigos CRC (véase más adelante) y otros códigos.

5.2.3 Comprobación de redundancia cíclica (CRC)

Una técnica detección de errores utilizada ampliamente en las redes de computadoras de hoy día esta basada en los **códigos de comprobación de redundancia cíclica (*CRC, Cyclic Redundancy Check*)**. Los códigos CRC también se conocen con el nombre de **códigos polinómicos**, dado que se puede ver la cadena de bits que hay que enviar como si fuera un polinomio cuyos coeficientes son los valores 0 y 1 de la cadena de bits, interpretándose las operaciones realizadas con la cadena de bits según la aritmética de polinomios.

Los códigos CRC operan de la forma siguiente. Considere la secuencia de datos de d bits, D, que el nodo emisor quiere transmitir al nodo receptor. El emisor y el receptor tienen que acordar primero un patrón de $r + 1$ bits, conocido como **generador**, que denominaremos con la letra G. Impondremos la condición de que el bit más significativo (el bit situado más a la izquierda) de G sea 1. La idea clave subyacente a los códigos CRC se muestra en la Figura 5.7. Para un determinada secuencia de datos, D, el emisor seleccionará r bits adicionales, R, y se los añadirá a D, de modo que el patrón de $d + r$ bits resultante (interpretado como un número binario) sea exactamente divisible por G (es decir, no tenga ningún resto) utilizando aritmética módulo 2. El proceso de comprobación de errores con los códigos CRC es, por tanto, muy simple: el receptor divide los $d + r$ bits recibidos entre G. Si el resto es distinto de cero, el receptor sabrá que se ha producido error; en caso contrario, se aceptarán los datos como correctos.

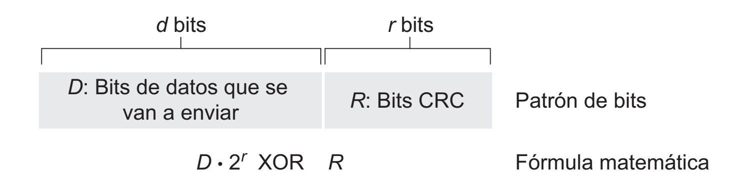


Figura 5.7 • CRC.

Todos los cálculos de los códigos CRC se realizan en aritmética módulo 2, sin ningún tipo de acarreo ni en las sumas ni en las restas. Esto quiere decir que la suma y la resta son idénticas, y que ambas son equivalentes a la operación OR-exclusiva (XOR) bit a bit de los operandos. Así, por ejemplo,

```
1011 XOR 0101 = 1110
1001 XOR 1101 = 0100
```

De forma similar, también tendremos que

```
1011 – 0101 = 1110
1001 – 1101 = 0100
```

La multiplicación y la división son iguales que en aritmética en base 2, excepto porque las sumas y restas necesarias se llevan a cabo sin acarreos. Como en la aritmética binaria ordinaria, la multiplicación por 2^k hace que un patrón de bits se desplace k posiciones hacia la izquierda. Por tanto, dados D y R, el valor $D \quad 2^r$ XOR R nos dará el patrón de $d + r$ bits mostrado en la Figura 5.7. Utilizaremos esta caracterización algebraica del patrón de $d + r$ bits de la Figura 5.7 en nuestras explicaciones.

Volvamos ahora a la pregunta crucial de cómo puede el emisor calcular R. Recuerde que queremos encontrar una secuencia R tal que exista n que cumpla

$$D \quad 2^r \text{ XOR } R = nG$$

Es decir, queremos seleccionar R tal que G divida a $D \quad 2^r$ XOR R sin que quede resto. Si aplicamos XOR (es decir, si sumamos en módulo 2 sin acarreo) R a ambos lados de la ecuación anterior, obtenemos

$$D \quad 2^r = nG \text{ XOR } R$$

Esta ecuación nos dice que si dividimos $D \quad 2^r$ entre G, el valor del resto será precisamente R. En otras palabras, podemos calcular R como

$$R = \text{resto } \frac{D \cdot 2^r}{G}$$

La Figura 5.8 ilustra este cálculo para el caso de $D = 101110$, $d = 6$, $G = 1001$ y $r = 3$. Los 9 bits transmitidos en este caso serán 101110 011. Puede comprobar estos cálculos por sí mismo y comprobar también que $D \quad 2^r = 101011 \quad G$ XOR R.

Se han definido estándares internacionales para generadores G de 8, 12, 16 y 32 bits. El estándar CRC-32 para 32 bits, que se ha adoptado en una serie de protocolos del IEEE para la capa de enlace, utiliza el generador

$$G_{\text{CRC-32}} = 100000100110000010001110110110111$$

Cada uno de los estándares de CRC puede detectar ráfagas de errores inferiores a $r + 1$ bits (esto significa que todos los errores de r bits consecutivos o menos serán detectados). Además, en las condiciones adecuadas, una ráfaga de longitud superior a $r + 1$ bits será detectada con una probabilidad de $1 - 0.5^r$. Asimismo, cada uno de los estándares CRC puede detectar cualquier número impar de errores de bit. En [Williams 1993] puede encontrar un análisis de la implementación de códigos CRC. La teoría subyacente a los códigos

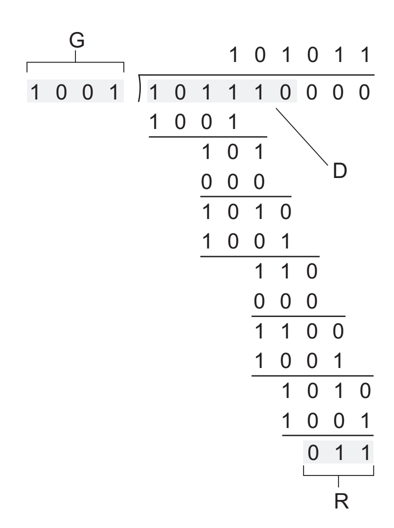


Figura 5.8 • Un ejemplo de cálculo de CRC.

CRC y a algunos códigos incluso más potentes cae fuera del alcance de este libro. El texto de [Schwartz 1980] proporciona una excelente introducción a este tema.

5.3 Protocolos de acceso múltiple

En la introducción de este capítulo hemos indicado que existen dos tipos de enlaces de red: enlaces punto a punto y enlaces de difusión (*broadcast*). Un **enlace punto a punto** está compuesto por un único emisor en un extremo del enlace y un único receptor en el otro extremo. Se han diseñado muchos protocolos de la capa de enlace para enlaces punto a punto; dos de esos protocolos, de los que hablaremos posteriormente en el capítulo, son el Protocolo punto a punto (PPP, *Point-to-Point Protocol*) y el protocolo de Control del enlace de datos de alto nivel (HDLC, *High-level Data Link Control*). El segundo tipo de enlace, un **enlace de difusión**, puede tener múltiples nodos emisores y receptores, todos conectados al mismo y único canal de difusión compartido. Utilizamos aquí el término *difusión* porque cuando un nodo transmite una trama, el canal se encarga de difundir esa trama y cada uno de los demás nodos recibe una copia. Ethernet y las redes LAN inalámbricas son ejemplos de tecnologías de difusión de la capa de enlace. En esta sección vamos a abstraernos momentáneamente de los protocolos específicos de la capa de enlace y vamos a examinar en primer lugar un problema de crucial importancia para esa capa: cómo coordinar el acceso de múltiples nodos emisores y receptores a un canal de difusión compartido, lo que se conoce con el nombre de **problema de acceso múltiple**. Los canales de difusión se suelen utilizar en las redes LAN, que son redes geográficamente concentradas en un único edificio (o un campus corporativo o universitario). Por ello, también examinaremos al final de esta sección cómo se utilizan los canales de acceso múltiple en las redes LAN.

Todos estamos familiarizados con la noción de transmisiones de difusión, ya que la televisión ha estado empleando este tipo de mecanismo desde que fuera inventada. Pero la televisión tradicional es una difusión en un único sentido (es decir, hay un nodo fijo que transmite a muchos nodos receptores), mientras que los nodos de un canal de difusión de

una red de computadoras pueden tanto enviar como recibir. Quizá una analogía más adecuada para un canal de difusión extraída del campo de las relaciones sociales sería un coctel en el que muchas personas se reúnen en una gran sala (el aire proporciona el medio de difusión) para hablar y escuchar. Una segunda analogía sería algo con lo que muchos lectores estarán familiarizados, una clase, en la que uno o más profesores y estudiantes comparten de forma similar el mismo y único medio de difusión. Un problema crucial en ambos escenarios es el de determinar quién es el que tiene derecho a la palabra (es decir, derecho a transmitir hacia el canal) y cuándo lo tiene. Las personas hemos llegado a desarrollar un conjunto elaborado de protocolos con el fin de compartir el canal de difusión:

"Dar a todo el mundo una oportunidad de hablar."

"No hablar hasta que te hablen."

"No monopolizar la conversación."

"Levantar la mano si se tiene una pregunta que plantear."

"No interrumpir cuando alguien está hablando."

"No quedarse dormido cuando alguien está hablando."

De forma similar, las redes de computadoras tienen protocolos (denominados **protocolos de acceso múltiple**) mediante los cuales los nodos se encargan de regular sus transmisiones al canal de difusión compartido. Como se muestra en la Figura 5.9, los protocolos de acceso múltiple son necesarios en una amplia variedad de escenarios de red, incluyendo las redes de área local tanto cableadas como inalámbricas y las redes de satélite. Aunque técnicamente cada nodo accede al canal de difusión a través de su adaptador, en esta sección nos referiremos con el término *nodo* a los dispositivos emisor y receptor. En la práctica, puede haber cientos o incluso miles de nodos comunicándose directamente a través de un canal de difusión.

Puesto que todos los nodos son capaces de transmitir tramas, podría darse el caso de que más de dos nodos transmitieran tramas al mismo tiempo. Cuando esto sucede, todos los nodos reciben varias tramas simultáneamente; es decir, las tramas transmitidas **colisionan** en todos los receptores. Normalmente, cuando se produce una colisión ninguno de los nodos receptores puede interpretar ninguna de las tramas transmitidas; en un cierto sentido, las señales de las tramas que han colisionado se entremezclan y no pueden separarse. Por tanto, todas las tramas implicadas en la colisión se pierden y el canal de difusión está desaprovechado durante el intervalo de colisión. Obviamente, si hay muchos nodos que quieren transmitir tramas de manera frecuente, muchas de las transmisiones provocarán colisiones y buena parte del ancho de banda del canal de difusión se desperdiciará.

Para poder garantizar que el canal de difusión realice un trabajo útil aun cuando haya múltiples nodos activos, es necesario coordinar de alguna manera las transmisiones de esos nodos activos. Este trabajo de coordinación es responsabilidad del protocolo de acceso múltiple. En los últimos 40 años se han escrito miles de artículos y cientos de tesis doctorales acerca de los protocolos de acceso múltiple; pueden encontrar en [Rom 1990] una panorámica bastante completa de los primeros 20 años de estas investigaciones. Hoy día, la investigación acerca de los protocolos de acceso múltiple continúa de manera activa debido a la continua aparición de nuevos tipos de enlaces y, en particular, de nuevos enlaces inalámbricos.

A lo largo de los años se han implementado docenas de protocolos de acceso múltiple utilizando diversas tecnologías de la capa de enlace. No obstante, podemos clasificar casi

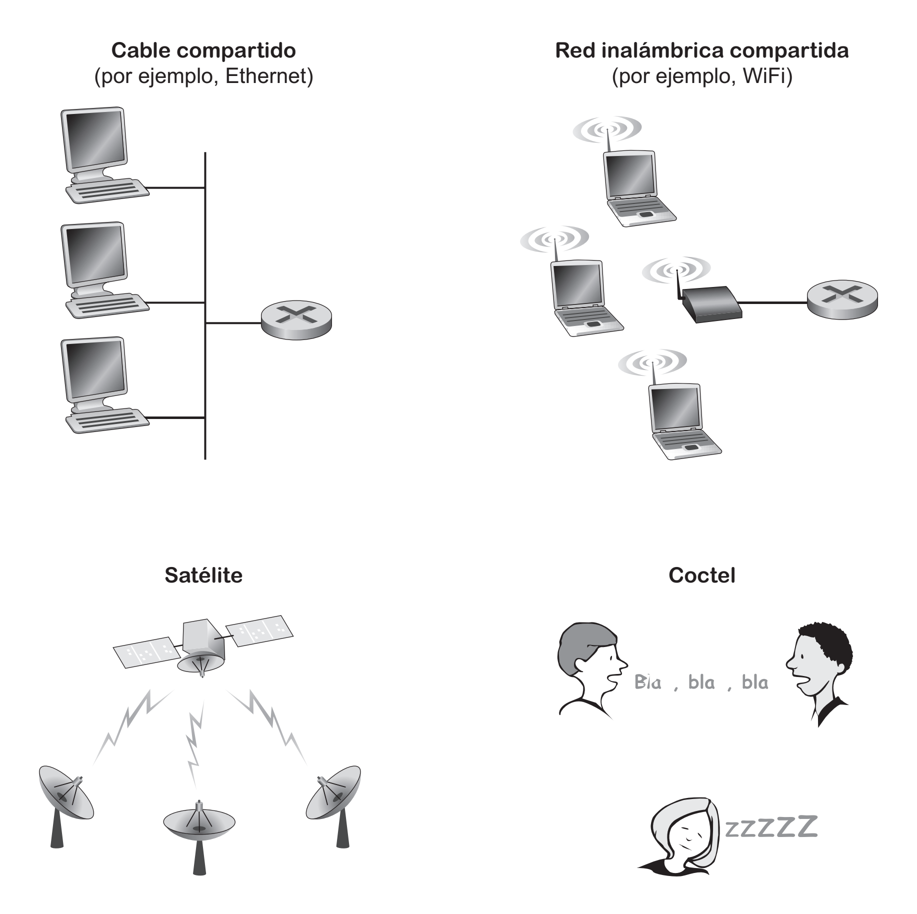


Figura 5.9 • Varios canales de acceso múltiple.

todos los protocolos de acceso múltiple en una de estas tres categorías: **protocolos de particionamiento del canal**, **protocolos de acceso aleatorio** y **protocolos de toma de turnos**. Hablaremos de estas tres categorías de protocolos de acceso multiple en las tres siguientes subsecciones.

Vamos a concluir esta introducción observando que, idealmente, un protocolo de acceso múltiple para un canal de difusión con una velocidad de *R* bits por segundo debería tener las siguientes características deseables:

1. Cuando sólo haya un nodo que tenga datos para enviar, a dicho nodo se le asignará una tasa de transferencia de *R* bps.
2. Cuando haya *M* nodos con datos para enviar, cada uno de esos nodos tendrá una tasa de transferencia de *R*/*M* bps. Esto no implica necesariamente que cada uno de los *M* nodos tenga siempre una tasa instantánea igual a *R*/*M,* sino más bien que cada nodo tendrá una tasa media de transmisión igual a *R*/*M* a lo largo de un intervalo de tiempo definido adecuadamente.
3. El protocolo será descentralizado; es decir, no habrá ningún nodo maestro que pueda actuar como punto único de fallo para la red.

4. El protocolo será simple, de modo que no sea costoso de implementar.

5.3.1 Protocolos de particionamiento del canal

Recuerde de nuestras explicaciones de la Sección 1.3 que la multiplexación por división en el tiempo (TDM) y la multiplexación por división de frecuencia (FDM) son dos técnicas que pueden utilizarse para particionar el ancho de banda de un canal de difusión entre todos los nodos que comparten el canal. Por ejemplo, suponga que el canal da soporte a N nodos y que la tasa de transmisión del canal es igual a R bps. TDM divide el tiempo en **marcos temporales** y luego subdivide cada marco temporal en N **particiones de tiempo**. (En inglés, el término *time frame,* marco de tiempo, de TDM no debe confundirse con la unidad de datos de la capa de enlace intercambiada entre las tarjetas adaptadoras de red del emisor y del receptor, que se denomina también *frame,* y que nosotros hemos denominado trama. En cualquier caso, para evitar confusiones en esta subsección nos referiremos a la unidad de datos de la capa de enlace intercambiada como paquete.) Cada partición de tiempo se asigna entonces a uno de los N nodos. Cada vez que un nodo tenga un paquete para enviar, transmite los bits del paquete durante su partición de tiempo asignada, dentro del marco TDM que se repite de forma cíclica. Normalmente, los tamaños de partición se eligen de modo que pueda transmitirse un único paquete durante la partición de tiempo asignada. La Figura 5.10 muestra un ejemplo simple de TDM con cuatro nodos. Si volvemos a nuestra analogía del coctel, un coctel regulado mediante TDM permitiría a uno de los participantes hablar durante un periodo fijo de tiempo, luego permitiría a otro participante hablar durante la misma cantidad de tiempo, y así sucesivamente. Una vez que todos hubieran tenido la oportunidad de hablar, el patrón se repetiría.

TDM resulta muy atractivo porque elimina las colisiones y es perfectamente equitativo. Cada nodo obtiene una tasa de transmisión dedicada igual a R/N bps durante cada marco temporal. Sin embargo, presenta dos importantes inconvenientes. En primer lugar, cada

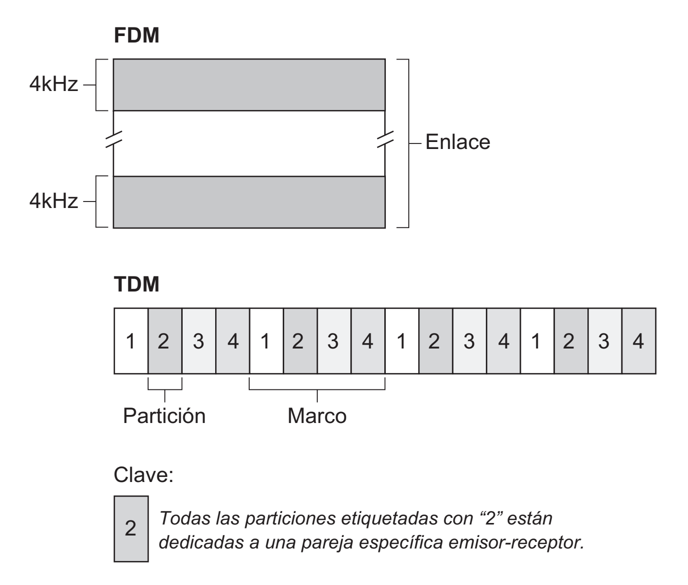


Figura 5.10 • Un ejemplo de TDM y FDM para cuatro nodos.

nodo está limitado a una tasa promedio de *R*/*N* bps aunque sea el único nodo que tiene paquetes para transmitir. El segundo inconveniente es que un nodo siempre tiene que esperar a que le llegue el turno dentro de la secuencia de transmisión; de nuevo, esa espera será obligatoria aún cuando sea el único nodo que tenga una trama que enviar. Imagine que esa situación se diera con un asistente al coctel que es el único que tiene algo que decir (imagine que nos encontráramos con la circunstancia, todavía más rara, que todo el mundo quiere escuchar lo que esa persona tiene que decir). Evidentemente, TDM sería un protocolo de acceso múltiple bastante inadecuado para ese coctel en concreto.

Mientras que TDM hace que los nodos compartan el canal de difusión a lo largo del tiempo, FDM divide el canal de *R* bps en diferentes frecuencias (cada una con un ancho de banda de *R*/*N*) y asigna cada frecuencia a cada uno de los *N* nodos. Así, FDM crea *N* canales más pequeños de *R*/*N* bps a partir de un único canal disponible mayor a *R* bps. FDM comparte con TDM tanto las ventajas como los inconvenientes. Evita las colisiones y divide el ancho de banda equitativamente entre los *N* nodos. Sin embargo, FDM también comparte una desventaja fundamental con TDM: cada nodo está limitado a un ancho de banda de *R*/*N*, incluso cuando sea el único nodo que tienen paquetes para enviar.

Un tercer protocolo de particionamiento del canal es el protocolo de **Acceso múltiple por división de código (CDMA, *Code Division Multiple Access*)**. Mientras que TDM y FDM asignan particiones de tiempo y frecuencias, respectivamente, a los nodos, CDMA asigna un *código* diferente a cada nodo. Cada nodo entonces utiliza su código único para codificar los bits de datos a enviar. Si se seleccionan los códigos cuidadosamente, las redes CDMA presentan la maravillosa característica de que los distintos nodos puede transmitir *simultáneamente* y conseguir que sus respectivos receptores decodifiquen correctamente los bits de datos codificados por el emisor (suponiendo que el receptor conozca el código utilizado por el emisor) aunque haya interferencias provocadas por las transmisiones realizadas por los otros nodos. CDMA se ha utilizado en sistemas militares durante algún tiempo (debido a su resistencia a las interferencias) y ahora se usa ampliamente en el mundo civil, en particular en la telefonía celular. Puesto que el uso de CDMA está tan estrechamente ligado a los canales inalámbricos, dejaremos las explicaciones acerca de los detalles técnicos de CDMA para el Capítulo 6. Por el momento, nos basta con saber que se pueden asignar códigos CDMA, al igual que particiones de tiempo en TDM y frecuencias en FDM a los usuarios del canal de acceso múltiple.

5.3.2 Protocolos de acceso aleatorio

La segunda clase general de protocolos de acceso múltiple son los protocolos de acceso aleatorio. En un protocolo de acceso aleatorio, cada nodo transmisor transmite siempre a la máxima velocidad del canal, que es *R* bps. Cuando se produce una colisión, cada uno de los nodos implicados en la colisión retransmite repetidamente su trama (es decir, su paquete) hasta que la trama consiga pasar sin sufrir colisiones. Pero cuando un nodo experimenta una colisión no retransmite necesariamente la trama de forma inmediata. *En lugar de ello, espera durante un tiempo aleatorio antes de retransmitir la trama.* Cada nodo implicado en una colisión selecciona un retardo aleatorio independientemente. Puesto que los retardos aleatorios son elegidos de forma independiente, es posible que uno de los nodos seleccione un retardo que sea suficientemente menor que los retardos de los otros nodos que han intervenido en la colisión, pudiendo así ser capaz de conseguir que su trama pase a través del canal sin experimentar una nueva colisión.

Existen docenas, si no centenares de protocolos de acceso aleatorio descritos en la literatura científica [Rom 1990; Bertsekas 1991]. En esta sección describiremos unos pocos de los protocolos de acceso aleatorio más comúnmente utilizados: los protocolos ALOHA [Abramson 1970; Abramson 1985] y los protocolos de Acceso múltiple con sondeo de portadora (CSMA, *Carrier Sense Multiple Access*) [Kleinrock 1975b]. Posteriormente, en la Sección 5.5, abordaremos los detalles de Ethernet [Metcalfe 1976], que es un protocolo CSMA muy popular y ampliamente difundido.

ALOHA con particiones

Comencemos nuestro estudio de los protocolos de acceso aleatorio con uno de los protocolos de este tipo más simples: el protocolo ALOHA con particiones. En nuestra descripción de ALOHA con particiones, haremos las siguientes suposiciones:

- Todas las tramas constan de exactamente L bits.
- El tiempo está dividido en particiones de L/R segundos (es decir, cada partición equivale al tiempo que se tarda en transmitir una trama).
- Los nodos comienzan a transmitir las tramas sólo al principio de las particiones.
- Los nodos están sincronizados, de modo que cada nodo sabe cuándo comienzan las particiones.
- Si dos o más tramas colisionan en una partición, entonces todos los nodos detectan la colisión incluso antes de que la partición termine.

Sea p una probabilidad, es decir, un número comprendido entre 0 y 1. El funcionamiento del protocolo ALOHA con particiones en cada nodo es simple:

- Cuando el nodo tiene una nueva trama que enviar, espera hasta el comienzo de la siguiente partición y transmite la trama completa dentro de la partición.
- Si no se produce una colisión, el nodo habrá transmitido correctamente su trama y por tanto no considerará la posibilidad de retransmitirla (el nodo puede preparar una nueva trama para su transmisión, si tiene una disponible).
- Si se produce una colisión, el nodo detecta la colisión antes de que la partición termine. El nodo retransmitirá su trama en cada partición posterior con una probabilidad p, hasta conseguir que la trama sea transmitida sin experimentar colisiones.

Al decir que se retransmite con probabilidad p, queremos decir que el nodo lleva a cabo en la práctica una especie de lanzamiento de una moneda trucada: si se sale cara, la trama se retransmite, lo cual sucede con probabilidad p; si sale cruz, se deja pasar la partición y se vuelve a lanzar la moneda para la partición siguiente, lo que ocurre con probabilidad $(1 - p)$. Todos los nodos implicados en la colisión "arrojan sus monedas" de forma independiente.

Puede parecer que el protocolo ALOHA con particiones tiene muchas ventajas. A diferencia de los mecanismos de particionamiento del canal, ALOHA con particiones permite a un nodo retransmitir continuamente a la velocidad máxima, R, cuando dicho nodo sea el único activo (decimos que un nodo está activo si tiene tramas que transmitir). ALOHA con particiones también es un protocolo altamente descentralizado, porque cada nodo detecta las colisiones y decide de forma independiente cuándo debe retransmitir (sin embargo,

ALOHA con particiones requiere que las particiones estén sincronizadas en los nodos; en breve analizaremos una versión no particionada del protocolo ALOHA, así como los protocolos CSMA , ninguno de los cuales requiere dicho tipo de sincronización). ALOHA con particiones es también un protocolo extremadamente simple.

ALOHA con particiones funciona bien cuando sólo hay un nodo activo, pero ¿qué eficiencia tiene cuando existen múltiples nodos activos? Son dos las posibles preocupaciones en lo que respecta a la eficiencia. En primer lugar, como se muestra en la Figura 5.11, cuando hay múltiples nodos activos una cierta fracción de las particiones experimentará colisiones y por tanto se "desperdiciará". La segunda preocupación es que otra fracción de las particiones estará *vacía* en aquellos casos en que todos los nodos activos se abstengan de transmitir, como resultado de la política probabilística de retransmisión. Las únicas particiones "no desperdiciadas" serán aquellas para las que haya exactamente un nodo transmitiendo. A las particiones en las que hay exactamente un nodo transmitiendo se las denomina **particiones con éxito**. La **eficiencia** de un protocolo de acceso múltiple con particiones se define como la fracción (calculada a largo plazo) de particiones con éxito cuando existe un gran número de nodos activos, cada uno de los cuales tiene siempre una gran cantidad de tramas que enviar. Observe que si no se utilizara ningún tipo de control de acceso y cada nodo intentara retransmitir inmediatamente después de cada colisión, la eficiencia sería cero. El protocolo ALOHA con particiones hace obviamente que la eficiencia aumente por encima de cero pero, ¿cuánto aumenta?

Vamos a esbozar el modo de determinar la eficiencia máxima del protocolo ALOHA con particiones. Para simplificar las cosas, vamos a modificar el protocolo ligeramente y a asumir que cada nodo trata de transmitir una trama en cada partición con probabilidad p. (Es decir, suponemos que todos los nodos tienen siempre una trama que enviar y que el nodo transmite con probabilidad p para las nuevas tramas, y no sólo para las tramas que ya hayan sufrido una colisión.) Suponga que existen N nodos. Entonces, la probabilidad de que una partición dada sea una partición de éxito es la probabilidad de que uno de los nodos trans-

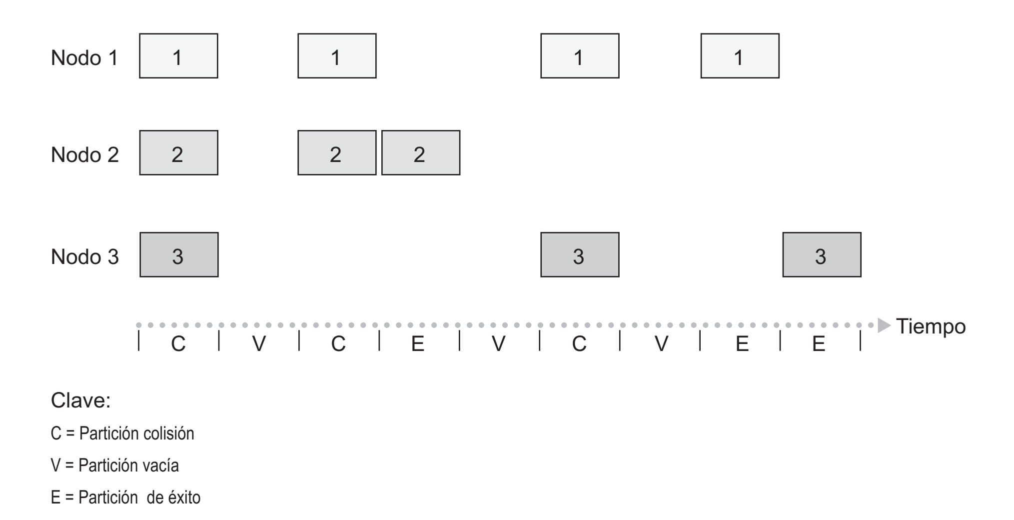


Figura 5.11 • Los nodos 1, 2 y 3 colisionan en la primera partición. El nodo 2 consigue tener éxito finalmente en la cuarta partición, el nodo 1 en la octava partición y el nodo 3 en la novena.

mita y de que los restantes $N - 1$ nodos no transmitan. La probabilidad de que un cierto nodo transmita es p; la probabilidad de que los demás nodos no transmitan es $(1 - p)^{N-1}$. Por tanto, la probabilidad de que un cierto nodo tenga éxito al transmitir será $p(1 - p)^{N-1}$. Puesto que hay N nodos, la probabilidad de que exactamente uno de los N nodos tenga éxito es $Np(1 - p)^{N-1}$.

Por tanto, cuando hay N nodos activos, la eficiencia del protocolo ALOHA con particiones es $Np(1 - p)^{N-1}$. Para obtener la eficiencia *máxima* para N nodos activos, tenemos que determinar el valor p^* que maximice esta expresión. (Consulte los problemas de repaso para ver un esbozo de estos cálculos.) Y para obtener la máxima eficiencia para un gran número de nodos activos, tomaremos el límite de $Np^*(1 - p^*)^{N-1}$ cuando N tiende a infinito. (Consulte de nuevo los problemas de repaso.) Después de realizar estos cálculos, se puede comprobar que la eficiencia máxima del protocolo está dada por $1/e$ 0,37. Es decir, cuando un gran número de nodos tienen muchas tramas que transmitir, entonces (como máximo) sólo el 37 por ciento de las particiones conseguirán transmitir la información con éxito. Por tanto, la velocidad de transmisión efectiva del canal no es R bps sino sólo 0,37 R bps. Un análisis similar muestra también que el 37 por ciento de las particiones quedarán vacías y el 26 por ciento sufrirán colisiones. ¡Imagínese al pobre administrador de la red que acaba de adquirir un sistema ALOHA con particiones a 100 Mbps y que esperaba poder utilizar la red para transmitir datos entre una gran cantidad de usuarios, con una velocidad agregada de, digamos, en torno a 80 Mbps! Aunque el canal es capaz de transmitir una determinada trama a la velocidad máxima del canal de 100 Mbps, a largo plazo, la tasa de transferencia efectiva de este canal será menor que 37 Mbps.

Aloha

El protocolo ALOHA con particiones requiere que todos los nodos sincronicen sus transmisiones para que éstas comiencen al principio de una partición. El primer protocolo ALOHA [Abramson 1970] era en realidad un protocolo no particionado y completamente descentralizado. En el protocolo ALOHA puro, cuando llega una trama (es decir, cuando se pasa un datagrama desde la capa de red en el nodo emisor) el nodo transmite inmediatamente la trama en su totalidad hacia el canal de difusión. Si una trama transmitida experimenta una colisión con una o más transmisiones de otros nodos, el nodo (después de transmitir completamente la trama que ha sufrido la colisión) retransmitirá la trama de forma inmediata con una probabilidad p. En caso contrario, el nodo esperará durante un tiempo equivalente al tiempo total de retransmisión de una trama. Después de esta espera, transmitirá la trama con probabilidad p, o esperará (permaneciendo inactivo) durante otro periodo de tiempo igual al tiempo de transmisión de una trama con una probabilidad $1 - p$.

Para determinar la eficiencia máxima del protocolo ALOHA puro vamos a centrarnos en un nodo individual. Haremos las mismas suposiciones que en nuestro análisis del protocolo ALOHA con particiones y tomaremos como unidad de tiempo el tiempo de transmisión de una trama. En cualquier instante, la probabilidad de que un nodo esté transmitiendo una trama será p. Suponga que esta trama comienza su transmisión en el instante t_0. Como se muestra en la Figura 5.12, para que esta trama pueda transmitirse con éxito ningún otro nodo puede iniciar su transmisión en el intervalo de tiempo $[t_0 - 1, t_0]$, ya que dicha transmisión se solaparía con el inicio de la transmisión de la trama del nodo i. La probabilidad de que todos los demás nodos no comiencen una transmisión en ese intervalo es $(1 - p)^{N-1}$. De forma similar, ningún otro nodo puede comenzar una transmisión mientras el nodo i está transmitiendo, ya que dicha transmisión se solaparía con la última parte de la transmisión

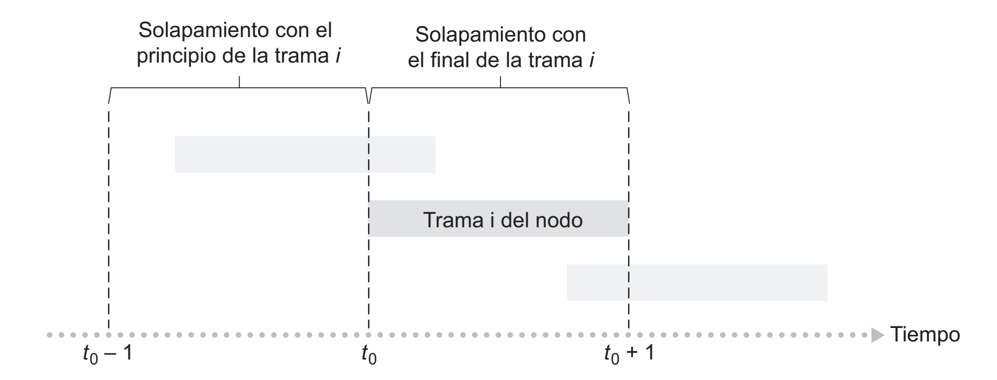


Figura 5.12 • Transmisiones que interfieren en el protocolo ALOHA puro.

del nodo *i*. La probabilidad de que todos los demás nodos no inicien una transmisión en este intervalo será también $(1 - p)^{N-1}$. Por tanto, la probabilidad de que un cierto nodo pueda transmitir con éxito será $p(1 - p)^{2(N-1)}$. Tomando límites como en el caso del protocolo ALOHA con particiones, encontramos que la máxima eficiencia del protocolo ALOHA puro es sólo de $1/(2e)$, que es exactamente la mitad que la del ALOHA con particiones. Por tanto, ese será el precio que habrá que pagar por disponer de un protocolo ALOHA completamente descentralizado.

HISTORIA

NORM ABRAMSON Y ALOHANET

Norm Abramson, doctor ingeniero, era un apasionado del surf y estaba también interesado en el tema de la conmutación de paquetes. Esa combinación de intereses le llevó a la universidad de Hawaii en 1969. Hawaii está compuesto por muchas islas montañosas, lo que hace difícil instalar y operar redes terrestres. Cuando no estaba practicando surf, Abramson se dedicada a pensar en cómo diseñar una red que realizara la conmutación de paquetes vía radio. La red que diseñó disponía de un host central y de varios nodos secundarios dispersos por las islas del archipiélago de Hawaii. La red tenía dos canales, cada uno de los cuales utilizaba una banda de frecuencia distinta. El canal de bajada difundía los paquetes desde el host central hacia los hosts secundarios, mientras que el de subida permitía enviar paquetes desde los hosts secundarios al host central. Además de enviar paquetes de información, el host central también enviaba a través del canal de bajada un mensaje de reconocimiento para cada uno de los paquetes recibidos desde los hosts secundarios.

Puesto que los hosts secundarios transmitían los paquetes de forma descentralizada, las colisiones en el canal de subida eran inevitables. Esta observación condujo a Abramson a desarrollar el protocolo ALOHA puro, descrito en este capítulo. En 1970, gracias a las aportaciones económicas de ARPA, Abramson conectó su red ALOHAnet con ARPAnet. El trabajo de Abramson es importante no sólo porque fue el primer ejemplo de red de paquetes vía radio, sino también porque sirvió de inspiración a Bob Metcalfe. Unos pocos años más tarde, Metcalfe modificó el protocolo para crear el protocolo CSMA/CD y las redes LAN Ethernet.

Acceso múltiple con sondeo de portadora (CSMA)

Tanto en el protocolo ALOHA puro como con particiones, la decisión de transmitir por parte de un nodo se toma independientemente de la actividad de los otros nodos conectados al canal de difusión. En particular, los nodos nunca prestan atención, en el momento de comenzar a transmitir, a si hay otros nodos transmitiendo ni tampoco dejan de transmitir si otro nodo comienza a interferir con su transmisión. En nuestra analogía del coctel, los protocolos ALOHA se parecen bastante a uno de esos invitados maleducados que continúan charlando independientemente de si hay otras personas haciendo uso de la palabra. Las personas disponemos de protocolos que nos permiten no sólo comportarnos de manera civilizada, sino también reducir la cantidad de tiempo desperdiciado "colisionando" unos con otros durante las conversaciones e incrementar así, como consecuencia, la cantidad de información que en nuestras conversaciones podemos intercambiar. Específicamente, hay dos reglas importantes de buena educación en las conversaciones que mantenemos los seres humanos:

- *Escuchar antes de hablar.* Si hay otra persona hablando, esperaremos hasta que haya terminado. En el mundo de las redes esto se denomina **sondeo de portadora**: cada nodo escucha el canal antes de transmitir. Si actualmente se está transmitiendo una trama de otro nodo por el canal, el nodo esperará un intervalo de tiempo aleatorio y luego volverá a sondear para ver si existe portadora en el canal. Si comprueba que el canal está inactivo, el nodo comenzará a transmitir su trama. En caso contrario, el nodo esperará otro intervalo aleatorio de tiempo y volverá a repetir este proceso.
- *Si alguien comienza a hablar al mismo tiempo, hay que dejar de hablar.* En el mundo de las redes esto se denomina **detección de colisiones**: un nodo que esté transmitiendo escuchará qué es lo que hay en el canal mientras dure la transmisión. Si detecta que otro nodo está transmitiendo una trama que interfiere la suya, dejará de transmitir y empleará algún tipo de protocolo para determinar cuándo debe volver a intentar transmitir de nuevo.

Estas dos reglas están integradas en la familia de protocolos de **Acceso múltiple con sondeo de portadora (CSMA, *Carrier Sense Multiple Access*)** y **CSMA con detección de colisiones (CSMA/CD)** [Kleinrock 1975b; Metcalfe 1976; Lam 1980; Rom 1990]. Se han propuesto muchas variantes de CSMA y CSMA/CD. Puede consultar esas referencias para conocer los detalles de estos protocolos. Estudiaremos en detalle el esquema de CSMA/CD utilizado en Ethernet en la Sección 5.5. Aquí consideraremos algunas de las características más importantes y fundamentales de CSMA y CSMA/CD.

La primera cuestión que podríamos plantearnos acerca de CSMA es por qué, si todos los nodos llevan a cabo un sondeo de portadora, se producen colisiones. Después de todo, los nodos se guardarán de transmitir cada vez que detecten que otro nodo está transmitiendo. La mejor forma de responder a esta pregunta es ilustrarla mediante diagramas espacio-tiempo [Molle 1987]. La Figura 5.13 muestra un diagrama espacio-tiempo de cuatro nodos (A, B, C, D) conectados a un bus lineal de difusión. El eje horizontal muestra la posición de cada nodo en el espacio y el eje vertical representa el tiempo.

En el instante t_0, el nodo B comprueba que el canal está inactivo, ya que no hay ningún nodo transmitiendo actualmente. Por tanto, el nodo B comenzará a transmitir propagándose sus bits en ambas direcciones a lo largo del medio de difusión. La propagación hacia abajo de los bits de B en la Figura 5.13 a lo largo del tiempo indica que hace falta un intervalo de tiempo distinto de cero para que los bits de B consigan propagarse (aunque lo hagan a una velocidad próxima a la de la luz) a lo largo de medio de difusión. En el instante t_1 ($t_1 > t_0$) el

nodo D tiene una trama que enviar. Aunque el nodo B actualmente está transmitiendo en el instante t_1, los bits que B está transmitiendo todavía no han alcanzado a D, por lo que D detectará que el canal está inactivo en t_1. De acuerdo con el protocolo CSMA, D por tanto comienza a transmitir su trama. Un corto intervalo de tiempo después, la transmisión de B comienza a interferir en D con la propia transmisión de D. A partir de la Figura 5.13 resulta evidente que el **retardo de propagación de canal** terminal a terminal de un canal de difusión (el tiempo que una señal tarda en propagarse de uno de los nodos a otro) desempeñará un papel fundamental a la hora de determinar el rendimiento del canal. Cuanto mayor sea este retardo de propagación, mayor será la probabilidad de que un nodo que efectúa el sondeo de portadora no sea capaz de detectar una transmisión que ya ha comenzado en otro nodo de la red.

En la Figura 5.13 los nodos no realizan una detección de colisiones; tanto B como D continúan transmitiendo sus tramas en su totalidad, aún cuando se haya producido una colisión. Cuando un nodo realiza una detección de colisiones deja de transmitir en cuanto detecta que se ha producido una colisión. La Figura 5.14 muestra el mismo escenario que la Figura 5.13 salvo porque los dos nodos abortan ahora su transmisión poco después de detectar que se ha producido una colisión. Evidentemente, añadir el mecanismo de detección de colisiones a un protocolo de acceso múltiple ayudará a incrementar el rendimiento del pro-

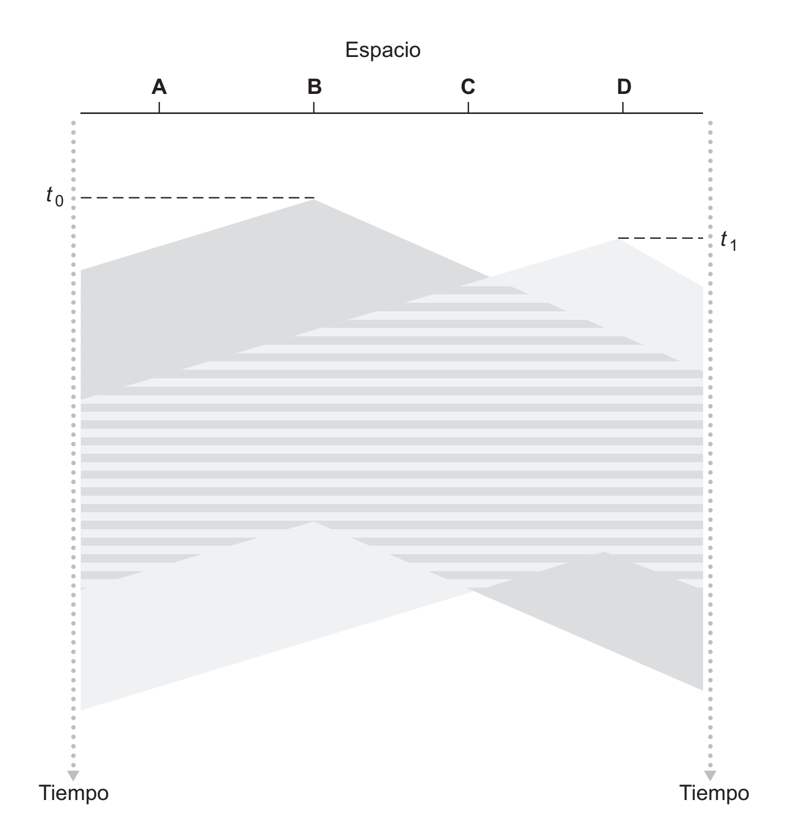


Figura 5.13 • Diagrama espacio-tiempo para dos nodos CSMA con transmisiones que entran en colisión.

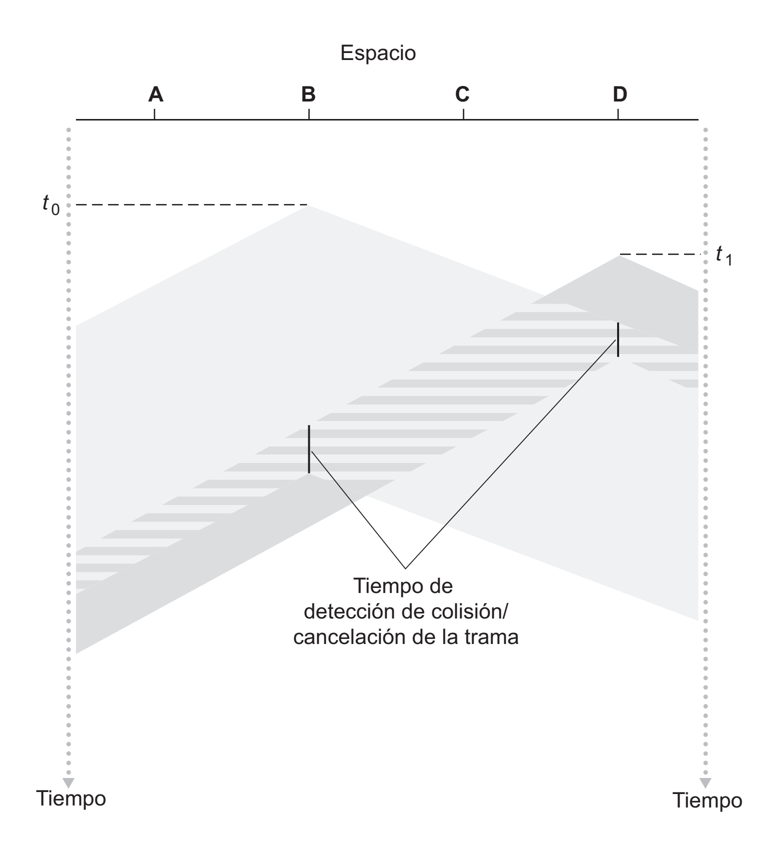


Figura 5.14 • CSMA con detección de colisiones.

tocolo, al no transmitirse una trama inútil, dañada (por interferencia con una trama de otro nodo) en su totalidad. El protocolo Ethernet que estudiaremos en la Sección 5.5 es un protocolo CSMA que utiliza detección de colisiones.

5.3.3 Protocolos de toma de turnos

Recuerde que dos propiedades deseables de un protocolo de acceso múltiple son: (1) cuando sólo haya un nodo activo, éste tendrá una tasa de transferencia de *R* bps y (2) cuando haya *M* nodos activos, entonces cada nodo activo dispondrá de una tasa de transferencia de aproximadamente *R*/*M* bps. Los protocolos ALOHA y CSMA presentan la primera de las propiedades, pero no la segunda. Esto ha servido de motivación para que una serie de investigadores creen otra clase de protocolos: los **protocolos de toma de turnos**. Al igual que con los protocolos de acceso aleatorio, existen docenas de protocolos de toma de turnos distintos y cada uno de estos protocolos tiene múltiples variantes. Aquí vamos hablar de dos de los protocolos más importantes de este tipo. El primero es el **protocolo de sondeo (*polling*)**. Este protocolo requiere que se designe a uno de los nodos como nodo maestro. El nodo maestro **sondea** a cada uno de los otros nodos a la manera de turno rotatorio (*round robin*). En particular, el nodo maestro envía primero un mensaje al nodo 1, diciéndole que puede transmitir hasta un cierto número máximo de tramas. Después de que el nodo 1 transmita una serie de tramas, el nodo maestro le dirá al nodo 2 que puede transmitir hasta el

máximo número de tramas. (El nodo maestro puede determinar cuándo un nodo ha terminado de enviar sus tramas observando la falta de señal en el canal.) El procedimiento continúa de esta forma ininterrumpidamente, encargándose el nodo maestro de sondear a cada uno de los otros nodos de forma cíclica.

El protocolo de sondeo elimina las colisiones y las particiones vacías que infectan los protocolos de acceso aleatorio. Esto permite que el mecanismo de sondeo consiga una eficiencia mucho mayor, aunque también presenta algunas desventajas. La primera es que el protocolo introduce un retardo de sondeo: el intervalo de tiempo requerido para indicarle a un nodo que puede transmitir. Por ejemplo, si sólo hay un nodo activo, entonces el nodo transmitirá a una velocidad menor que *R* bps, ya que el nodo maestro deberá sondear a cada uno de los nodos inactivos por turno cada vez que el nodo activo haya terminado de enviar su número máximo de tramas. El segundo inconveniente, que puede ser más grave, es que si el nodo maestro falla, entonces todo el canal dejará de estar operativo. El protocolo 802.15 y el protocolo Bluetooth que estudiaremos en la Sección 6.3 son ejemplos de protocolos de sondeo.

El segundo protocolo de toma de turnos es el **protocolo de paso de testigo**. En este protocolo no existe ningún nodo maestro; en su lugar hay una trama de pequeño tamaño y de propósito especial conocida con el nombre de **testigo** (*token*) que va siendo intercambiada entre los nodos en un determinado orden fijo. Por ejemplo, puede que el nodo 1 envíe el testigo siempre al nodo 2, el nodo 2 al nodo 3 y el nodo *N* al nodo 1. Cuando un nodo recibe el testigo, lo retiene si dispone de alguna trama para transmitir; en caso contrario, reenvía inmediatamente el testigo al siguiente nodo. Si un nodo tiene tramas que transmitir cuando recibe el testigo, envía una trama detrás de otra, hasta el número máximo de tramas permitido y luego reenvía el testigo al siguiente nodo. El mecanismo de paso de testigo es descentralizado y extremadamente eficiente, aunque también tiene sus propios problemas. Por ejemplo, el fallo de un nodo puede hacer que todo el canal quede inutilizable, o si un nodo se olvidara accidentalmente de liberar el testigo, entonces sería necesario invocar algún procedimiento de recuperación para hacer que el testigo vuelva a circular. A lo largo de los años se han desarrollado muchos protocolos de paso de testigo y cada uno de ellos tuvo que preocuparse de resolver estos problemas, así como algunos otros problemas complicados; haremos mención de estos protocolos, FDDI e IEEE 802.5, en la siguiente sección.

5.3.4 Redes de área local (LAN)

Los protocolos de acceso múltiple se utilizan junto con muchos tipos distintos de canales de difusión. Se han utilizado para canales de satélite e inalámbricos, cuyos nodos transmiten a través de un espectro de frecuencia común. Actualmente se utilizan en el canal de subida para el acceso a Internet por cable (véase la Sección 1.2), y también se utilizan ampliamente en las redes de área local (LAN).

Recuerde que una LAN es una red de computadoras concentrada en un área geográfica, como por ejemplo en un edificio o en un campus universitario. Cuando un usuario accede a Internet desde un campus universitario o corporativo, el acceso casi siempre se realiza a través de una LAN; específicamente, el acceso se lleva a cabo desde un host a la LAN, desde ésta a un router y desde el router a Internet, como se muestra en la Figura 5.15. La velocidad de transmisión, *R,* de la mayoría de las redes LAN es muy alta. Incluso a principios de la década de 1980, las redes LAN a 10 Mbps eran bastante comunes; hoy día, resultan comunes las redes LAN de 100 Mbps y 1 Gbps y ya hay disponibles redes LAN a 10 Gbps.

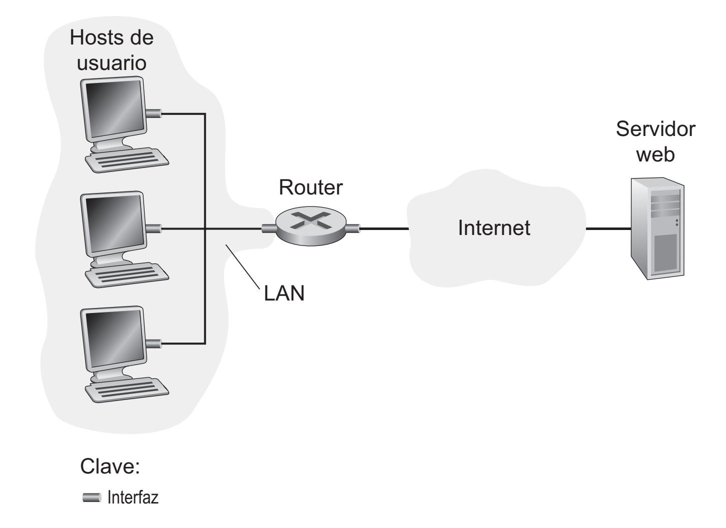


Figura 5.15 • Hosts de usuario accediendo a un servidor web de Internet a través de una LAN, estando compuesto el canal de difusión entre un host de usuario y el router por un único enlace.

En la década de 1980 y a principios de la de 1990 había dos clases de tecnologías LAN muy populares en los entornos empresariales. La primera clase era la de las redes LAN Ethernet (también conocidas como redes LAN 802.3 [IEEE 802.3 2009]), que están basadas en un mecanismo de acceso aleatorio. La segunda clase de tecnologías LAN estaba compuesta por las tecnologías de paso de testigo, incluyendo la tecnología **token ring** (también conocida como IEEE 802.5 [IEEE 802.5 2009]) y la **Interfaz de datos distribuidos para fibra (FDDI, *Fiber Distributed Data Interface*)** [Jain 1994]. Dado que vamos a analizar las tecnologías Ethernet con un cierto grado de detalle en la Sección 5.5, centraremos aquí nuestras explicaciones en las redes LAN de paso de testigo. Nuestra exposición acerca de las tecnologías de paso de testigo es intencionadamente breve, porque la implacable competencia de las redes Ethernet ha hecho que estas tecnologías prácticamente se hayan extinguido. De todos modos, para poder proporcionar ejemplos de tecnologías de paso de testigo y para poder disponer de una cierta perspectiva histórica resulta útil decir unas cuantas palabras acerca de las redes de paso de testigo.

En una LAN token ring, los N nodos de la LAN (hosts y routers) están conectados en anillo mediante una serie de enlaces directos. La topología en anillo de paso de testigo define el orden en que el testigo se pasa de un nodo a otro. Cuando un nodo obtiene el testigo y envía una trama, la trama se propaga alrededor del anillo completo, creando así un canal de difusión virtual. El nodo de destino lee la trama a partir del medio de la capa de enlace, en el momento que la trama se propaga a su través. El nodo que envía la trama tiene la responsabilidad de eliminar la trama del anillo. FDDI se diseñó para redes LAN geográficamente más amplias, incluyendo las **redes de área metropolitana (MAN, *Metropolitan Area Network*)**. Para las redes LAN geográficamente amplias (distribuidas a lo largo de varios kilómetros) es poco eficiente dejar que una trama se propague de vuelta hacia el nodo emisor después de haber sobrepasado el nodo de destino. FDDI hace que el nodo de destino elimine la trama del anillo. (Hablando en sentido estricto, FDDI no es por tanto un canal de difusión puro, ya que no todos los nodos reciben las tramas transmitidas.)

5.4 Direccionamiento de la capa de enlace

Los nodos (es decir, los hosts y los routers) tienen direcciones de la capa de enlace. Esto podría parecer sorprendente si recordamos del Capítulo 4 que los nodos tienen también direcciones de la capa de red. Es posible que el lector se esté preguntando para qué necesitamos disponer de direcciones tanto en la capa de red como en la de enlace. Además de describir la sintaxis y la función de las direcciones de la capa de enlace, en esta sección confiamos en arrojar algo de luz sobre las razones por las que resulta útil emplear las dos capas de direcciones y, de hecho, veremos que esto es indispensable. También nos ocuparemos del Protocolo de resolución de direcciones (ARP, *Address Resolution Protocol*), que proporciona un mecanismo para traducir las direcciones IP en direcciones de la capa de enlace.

5.4.1 Direcciones MAC

En realidad, no son los nodos (es decir, los hosts o routers) los que tienen asignadas direcciones de la capa de enlace, sino que las direcciones de la capa de enlace se asignan a los adaptadores instalados en cada nodo. Esto se ilustra en la Figura 5.16. A las direcciones de la capa de enlace se las denomina de diversas formas como **dirección LAN**, **dirección física** o **dirección MAC**. Dado que el término dirección MAC parece ser el más popular, en lo sucesivo nos referiremos a las direcciones de la capa de enlace utilizando dicho término. En la mayoría de las redes LAN (incluyendo las redes Ethernet y las LAN inalámbricas 802.11), la dirección MAC tiene 6 bytes de longitud, lo que nos da 2^{48} posibles direcciones MAC. Como se muestra en la Figura 5.16, estas direcciones de 6 bytes suelen expresarse en notación hexadecimal, indicándose cada byte de la dirección mediante una pareja de números hexadecimales. Aunque las direcciones MAC se diseñaron para ser permanentes, hoy día es posible modificar la dirección MAC de un adaptador mediante un software apropiado. Sin embargo, en el resto de esta sección supondremos que la dirección MAC de un adaptador es fija.

Una propiedad interesante de las direcciones MAC es que nunca puede haber dos adaptadores con la misma dirección. Esto puede parecer sorprendente, dado que los adaptadores son fabricados por muchas compañías distintas en muchos países diferentes. ¿Cómo puede una empresa fabricante de adaptadores de Taiwan estar segura de que está utilizando un conjunto diferente de direcciones del que emplea otra empresa que fabrica adaptadores en Bélgica? La respuesta es que el IEEE se encarga de gestionar el espacio de direcciones MAC. En particular, cuando una empresa quiere fabricar adaptadores, compra por un precio fijado una parte del espacio de direcciones compuesto por 2^{24} direcciones. IEEE asigna el fragmento de 2^{24} direcciones fijando los primeros 24 bits de una dirección MAC y dejando que la empresa diseñe combinaciones únicas de los últimos 24 bits para cada adaptador.

La dirección MAC de un adaptador tiene una estructura plana (en oposición a una estructura jerárquica) y nunca varía independientemente de a dónde se lleve el adaptador. Una computadora portátil con una tarjeta Ethernet siempre tendrá la misma dirección MAC, independientemente de dónde se utilice esa computadora. Una PDA con una interfaz 802.11 tendrá siempre también la misma dirección MAC, independientemente de dónde la llevemos. Recuerde que, por contraste, las direcciones IP tienen una estructura jerárquica (es decir, una parte de red y una parte de host) y que es necesario modificar la dirección IP de un nodo cuando el host se mueve, es decir, cuando cambia la red a la que el host está conectado. La dirección MAC de un adaptador es análoga al número del carnet de identidad o de

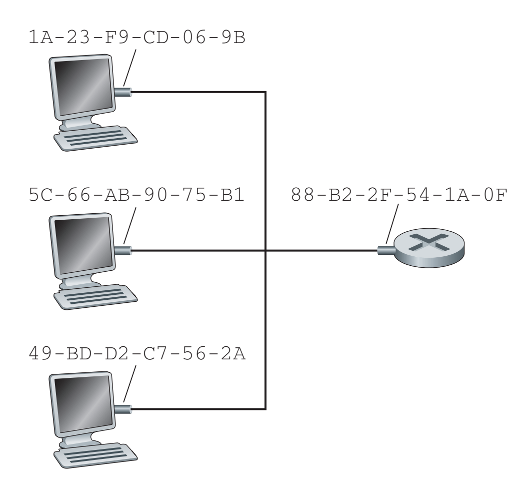


Figura 5.16 • Cada adaptador conectado a una LAN dispone de una dirección MAC única.

la seguridad social de una persona, que también tiene una estructura de direccionamiento plana que no cambia independientemente de a dónde se vaya a vivir esa persona. Un dirección IP, por su parte, sería análoga a la dirección postal de una personas, que es jerárquica y que debe modificarse cada vez que una persona cambia de domicilio. Al igual que para las personas resulta útil disponer tanto de una dirección postal como de un número de la seguridad social, también para los nodos resulta útil disponer de una dirección de la capa de red y de una dirección MAC.

Como hemos descrito al principio de esta sección, cuando un adaptador de un emisor quiere enviar una trama a otro adaptador de destino, inserta la dirección MAC del de destino en la trama y luego la envía a través de la red LAN. Si la red LAN es una LAN de difusión (como por ejemplo, 802.11 o Ethernet), la trama será recibida y procesada por todos los demás adaptadores de la LAN. En particular, cada adaptador que reciba la trama comprobará si la dirección MAC de destino contenida en la trama se corresponde con su propia dirección MAC. Si existe una correspondencia, el adaptador extraerá el datagrama incluido en la trama y lo pasará hacia arriba por la pila de protocolos para entregárselo a su nodo padre. Si no hay una correspondencia entre ambas direcciones, el adaptador descarta la trama, sin pasar el datagrama de la capa de red hacia arriba por la pila de protocolos. De este modo, sólo el nodo de destino será interrumpido cuando se reciba la trama.

Sin embargo, en ocasiones un adaptador de un emisor *sí que quiere* que todos los demás adaptadores de la LAN reciban y *procesen* la trama que va a enviar. En este caso, el adaptador emisor inserta una **dirección de difusión** MAC especial en el campo de la dirección de destino de la trama. Para las redes LAN que utilizan direcciones de 6 bytes (como las LAN Ethernet y de paso de testigo), la dirección de difusión es una cadena compuesta por 48 unos (1) consecutivos (es decir, FF-FF-FF-FF-FF-FF en notación hexadecimal).

5.4.2 Protocolo de resolución de direcciones (ARP)

Dado que existen tanto direcciones de la capa de red (por ejemplo, direcciones IP de Internet) como direcciones de la capa de enlace (es decir, direcciones MAC), surge la necesidad



MANTENER LAS CAPAS INDEPENDIENTES

Existen varias razones por las que los nodos tienen direcciones MAC además de las direcciones de la capa de red. En primer lugar, las redes LAN están diseñadas para protocolos arbitrarios de la capa de red, no sólo para IP e Internet. Si los adaptadores de red tuvieran asignadas direcciones IP en lugar de direcciones MAC "neutras", entonces no podrían dar soporte fácilmente a otros protocolos de la capa de red (como por ejemplo, IPX o DECnet). En segundo lugar, si los adaptadores utilizaran direcciones de la capa de red en lugar de direcciones MAC, la dirección de la capa de red se tendría que almacenar en la memoria RAM del adaptador y tendría que reconfigurarse cada vez que el adaptador se moviera (o se encendiera). Otra opción sería no utilizar ninguna dirección en los adaptadores y que cada adaptador pasara los datos (normalmente, un datagrama IP) de cada trama recibida hacia arriba por la pila de protocolos. La capa de red podría entonces comprobar si existe una coincidencia con la dirección de la capa de red. Un problema que surge con esta opción es que el host sería interrumpido por cada trama enviada a través de la LAN, incluyendo aquellas tramas que estuvieran destinadas a otros nodos de la misma LAN de difusión. En resumen, con el fin de que las capas sean independientes en gran medida en una arquitectura de red, las distintas capas necesitan disponer de su propio esquema de direccionamiento. Hasta ahora hemos hablado de tres tipos de direcciones: nombres de host para la capa de aplicación, direcciones IP para la capa de red y direcciones MAC para la capa de enlace.

de una traducción entre ellas. En Internet, esta tarea la lleva a cabo el **Protocolo de resolución de direcciones (ARP, *Address Resolution Protocol*)** [RFC 826].

Para comprender la necesidad de un protocolo como ARP, considere la red mostrada en la Figura 5.17. En este sencillo ejemplo, cada nodo tiene una dirección IP única y el adaptador de cada nodo tiene una dirección MAC única. Como siempre, las direcciones IP se muestran en notación decimal con punto y las direcciones MAC en notación hexadecimal. Supongamos ahora que el nodo con la dirección IP 222.222.222.220 desea enviar un datagrama IP al nodo 222.222.222.222. En este ejemplo, tanto el nodo de origen como el destino se encuentran en la misma red (LAN) en el sentido de direccionamiento expresado en la Sección 4.4.2. Para enviar un datagrama, el nodo de origen tiene que proporcionar a su adaptador no sólo el datagrama IP sino también la dirección MAC del nodo de destino 222.222.222.222. El adaptador del nodo emisor construirá entonces una trama de la capa de enlace que contendrá la dirección MAC del nodo destino y enviará la trama a la red LAN.

La pregunta fundamental en esta sección es: ¿Cómo determina el nodo de origen la dirección MAC del nodo de destino con la dirección IP 222.222.222.222? Como posiblemente ya haya adivinado, utiliza el protocolo ARP. Un módulo ARP en el nodo emisor toma como entrada cualquier dirección IP de la misma LAN y devuelve la dirección MAC correspondiente. En nuestro ejemplo, el nodo emisor 222.222.222.220 proporciona a su módulo ARP la dirección IP 222.222.222.222 y el módulo ARP devuelve la correspondiente dirección MAC 49-BD-D2-C7-56-2A.

Vemos por tanto que ARP resuelve una dirección IP en una dirección MAC. En muchos sentidos, esto es análogo a DNS (estudiado en la Sección 2.5), que resuelve nombres de host en direcciones IP. Sin embargo, una diferencia importante entre los dos resolvedores es que DNS resuelve nombres de host para hosts ubicados en cualquier lugar de Internet, mientras

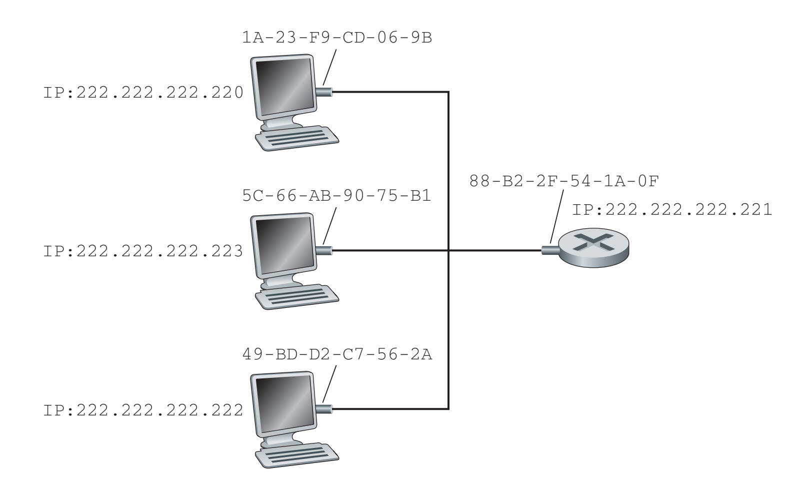


Figura 5.17 • Cada nodo de una LAN tiene una dirección IP y cada adaptador de un nodo tiene una dirección MAC.

que ARP resuelve direcciones IP sólo para los nodos de una misma subred. Si un nodo situado en California intentara utilizar ARP para resolver la dirección IP de un nodo en Mississippi, ARP devolvería un error.

Ahora que hemos explicado lo que hace ARP, vamos a ver cómo lo hace. Cada nodo (host o router) tiene en su memoria una **tabla ARP**, que contiene las correspondencias entre las direcciones IP y las direcciones MAC. La Figura 5.18 muestra el aspecto que puede tener la tabla ARP del nodo 222.222.222.220. La tabla ARP también contiene un valor de tiempo de vida (TTL), que indica cuándo se eliminará cada correspondencia de la tabla. Observe que la tabla no necesariamente contiene una entrada para cada nodo de la subred; algunos nodos pueden haber tenido entradas que han caducado, mientras que otros puede que nunca hayan tenido una entrada en la tabla. El tiempo típico de caducidad de una entrada es de 20 minutos desde el momento que se incluye la entrada en la tabla ARP.

Suponga ahora que el nodo 222.222.222.220 quiere enviar un datagrama con direccionamiento IP a otro nodo de dicha subred. El nodo emisor necesita obtener la dirección MAC del nodo de destino dada la dirección IP de dicho nodo. Esta tarea es fácil si la tabla ARP del nodo emisor tiene una entrada para nodo de destino. Pero, ¿qué ocurre si la tabla ARP no contiene actualmente una entrada para el nodo de destino? En particular, suponga que el nodo 222.222.222.220 desea enviar un datagrama al nodo 222.222.222. 222. En este caso, el nodo emisor utiliza el protocolo ARP para resolver la dirección. En primer lugar, el nodo emisor construye un paquete especial denominado **paquete ARP**. Un paquete ARP contiene varios campos, incluyendo las direcciones MAC e

Dirección IP	Dirección MAC	TTL
222.222.222.221	88-B2-2F-54-1A-0F	13:45:00
222.222.222.223	5C-66-AB-90-75-B1	13:52:00

Figura 5.18 • Una posible tabla ARP en el nodo 222.222.222.220.

IP del emisor y el receptor. Los paquetes de consulta y de respuesta ARP tienen el mismo formato. El propósito del paquete de consulta ARP es consultar a todos los demás nodos de la subred con el fin de determinar la dirección MAC correspondiente a la dirección IP que está resolviendo.

Volvamos a nuestro ejemplo. El nodo 222.222.222.220 pasa un paquete de consulta ARP al adaptador junto con una indicación de que el adaptador enviará el paquete a la dirección de difusión MAC, FF-FF-FF-FF-FF-FF. El adaptador encapsula el paquete ARP en una trama de la capa de enlace, utiliza la dirección de difusión para la dirección de destino de la trama y la transmite a la subred. Recuerde la analogía del número de la seguridad social y la dirección postal: una consulta ARP es equivalente a una persona gritando en una sala abarrotada de cubículos de alguna empresa (por ejemplo, CualquierEmpresa): "¿Cuál es el número de la seguridad social de la persona cuya dirección postal es Cubículo 13, Sala 112, CualquierEmpresa, Palo Alto, California?" La trama que contiene la consulta ARP es recibida por todos los demás adaptadores existentes en la subred y (a causa de la dirección de difusión) cada adaptador pasa la consulta ARP contenida en la trama al módulo ARP de dicho nodo. Cada nodo realiza una comprobación para ver si su dirección IP se corresponde con la dirección IP de destino del paquete ARP. El único nodo en el que se produzca la coincidencia devolverá al nodo que ha realizado la consulta una respuesta ARP con la correspondencia deseada. El nodo que ha realizado la consulta 222.222.222.220 podrá entonces actualizar su tabla ARP y enviar su datagrama IP, encapsulado dentro de una trama de la capa de enlace cuya dirección de destino MAC es la del nodo que ha contestado a la anterior consulta ARP.

Hay un par de cosas interesantes que comentar acerca del protocolo ARP. En primer lugar, el mensaje ARP de consulta se envía dentro de una trama de difusión, mientras que el mensaje ARP de respuesta se envían dentro de una trama estándar. Antes de continuar leyendo debería pararse a pensar por qué esto es así. En segundo lugar, ARP es un protocolo plug-and-play; es decir, la tabla ARP de un nodo se construye automáticamente (no tiene que ser configurada por el administrador del sistema). Y si un nodo está desconectado de la subred, su entrada finalmente se elimina de las tablas de los restantes nodos de la subred.

A menudo, los estudiantes se preguntan si ARP es un protocolo de la capa de enlace o un protocolo de la capa de red. Como hemos visto, un paquete ARP se encapsula dentro de una trama de la capa de enlace y así se sitúa, arquitectónicamente, encima de la capa de enlace. Sin embargo, un paquete ARP dispone de campos que contienen direcciones de la capa de enlace, por lo que se podría decir que es un protocolo de la capa de enlace, pero también contiene direcciones de la capa de red y, por tanto, podría también argumentarse que es un protocolo de la capa de red. En último término, probablemente ARP sea considerado un protocolo que se encuentra a caballo entre las capas de enlace y de red (no se ajusta limpiamente a la pila de protocolos en capas simples que hemos estudiado en el Capítulo 1). ¡Tales son las complejidades de los protocolos del mundo real!

Envío de un datagrama a un nodo fuera de la subred

Ahora debería tener claro cómo funciona ARP cuando un nodo desea enviar un datagrama a otro nodo que se encuentra en *la misma subred.* Pasemos entonces a un situación algo más complicada en la que un nodo de una subred desea enviar un datagrama de la capa de red a un nodo que está *fuera de la subred* (es decir, a través de un router a otra subred). Estudiemos esta situación en el contexto de la Figura 5.19, que muestra una red simple que consta de dos subredes interconectadas mediante un router.

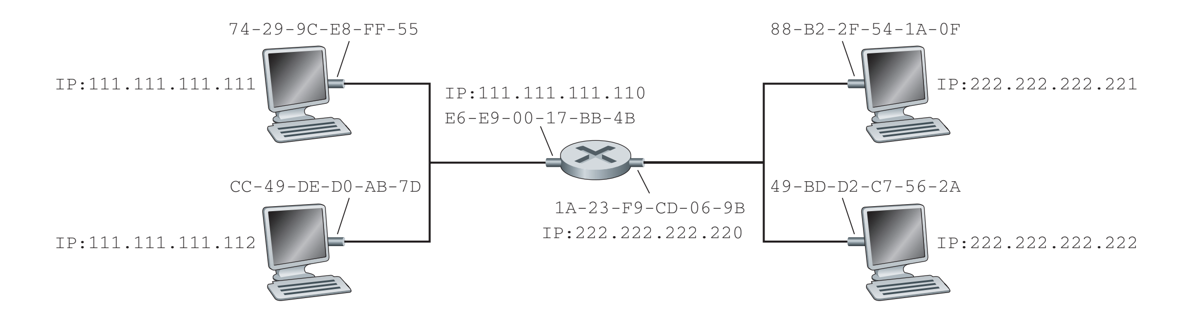


Figura 5.19 • Dos subredes interconectadas mediante un router.

Hay varias cuestiones interesantes que destacar en la Figura 5.19. En primer lugar, existen dos tipos de nodos: hosts y routers. Cada host tiene exactamente una dirección IP y un adaptador. Pero, como se ha visto en el Capítulo 4, un router tiene una dirección IP para *cada una* de sus interfaces. Para cada interfaz de router existe también un módulo ARP (en el router) y un adaptador. Dado que el router de la Figura 5.19 tiene dos interfaces, tendrá dos direcciones IP, dos módulos ARP y dos adaptadores. Por supuesto, cada adaptador de la red tiene su propia dirección MAC.

Fíjese también en que la Subred 1 tiene la dirección de red 111.111.111/24 y que la Subred 2 tiene la dirección de red 222.222.222/24. Así, todas las interfaces conectadas a la Subred 1 tienen direcciones de la forma 111.111.111.xxx y todas las interfaces conectadas a la Subred 2 tienen direcciones de la forma 222.222.222.xxx.

Examinemos ahora cómo un host de la Subred 1 enviaría un datagrama a un host de la Subred 2. Específicamente, suponga que el host 111.111.111.111 desea enviar un datagrama IP a un host 222.222.222.222. Como es habitual, el host emisor pasa el datagrama a su adaptador. Pero el host emisor también tiene que indicar a su adaptador una dirección MAC de destino apropiada. ¿Qué dirección MAC debería utilizar el adaptador? Una posibilidad sería probar si la dirección MAC apropiada es la del adaptador para el host 222.222.222.222, es decir, 49-BD-D2-C7-56-2A. Sin embargo, esta suposición resultaría errónea. Si el adaptador del emisor utilizara dicha dirección MAC, entonces ninguno de los adaptadores de la Subred 1 se molestaría en pasar el datagrama IP a su capa de red, ya que la dirección de destino de la trama no coincidiría con al dirección MAC de ningún adaptador de la Subred 1. El datagrama terminaría muriendo e iría al cielo de los datagramas.

Si nos fijamos en la Figura 5.19 vemos que para que un datagrama vaya desde 111.111.111.111 a un nodo de la Subred 2, el datagrama tiene en primer lugar que ser enviado a la interfaz de router 111.111.111.110, que es la dirección IP del router del primer salto en el camino hacia su destino final. Por tanto, la dirección MAC apropiada para la trama es la dirección del adaptador de la interfaz de router 111.111.111.110, es decir, E6-E9-00-17-BB-4B. ¿Cómo adquiere el host de emisor la dirección MAC para la dirección 111.111.111.110? ¡Por supuesto, utilizando ARP! Una vez que el adaptador del emisor tiene esta dirección MAC, crea una trama (que contiene el datagrama direccionado a 222.222.22.22) y envía la trama hacia la Subred 1. El adaptador del router de la Subred 1 ve que la trama de la capa de enlace se dirige hacia él y, por tanto, pasa la trama a la capa de red del router. ¡Estupendo! El diagrama IP se ha transmitido con éxito desde el host de origen al router. Pero todavía no hemos terminado. Todavía nos queda llevar el datagrama desde el router al destino. Ahora el router tiene que determinar la interfaz correcta a la que

el datagrama será reenviado. Como se ha explicado en el Capítulo 4, esto se hace consultando la tabla de reenvío del router. La tabla de reenvío indica al router que el datagrama es reenvíado a través de la interfaz de router 222.222.222.220. Esta interfaz entonces pasa el datagrama a su adaptador, que encapsula el datagrama en una nueva trama y la envía a la subred 2. Esta vez, la dirección MAC de destino de la trama es la dirección MAC del destino final. Pero, ¿cómo obtiene el router esta dirección MAC de destino? ¡Por supuesto, de ARP!

ARP para Ethernet está definido en el documento RFC 826. En el tutorial de TCP/IP, RFC 1180, se proporciona una introducción a ARP. Exploraremos en más detalle ARP en los problemas de repaso.

5.5 Ethernet

Ethernet ha avanzado mucho en el mercado de las redes LAN cableadas. En la década de 1980 y a principios de la década de 1990, Ethernet se enfrentó a muchos de los desafíos de otras tecnologías LAN, como Token Ring, FDDI y ATM. Algunas de estas otras tecnologías tuvieron éxito y captaron parte del mercado de las redes LAN durante unos pocos años. Pero, desde su aparición a mediados de la década de 1970, Ethernet ha continuado evolucionando y creciendo, y se ha mantenido en una posición dominante. Actualmente, Ethernet es de lejos la tecnología para redes LAN cableadas predominante y, probablemente, se mantendrá ahí en el futuro. Puede decirse que Ethernet ha sido a las redes de área local lo que Internet a las redes globales.

Existen muchas razones por las que Ethernet ha tenido éxito. En primer lugar, Ethernet fue la primera LAN de alta velocidad ampliamente implantada. Puesto que fue implantada muy pronto, los administradores de redes están extremadamente familiarizados con Ethernet (conocen sus grandezas y sus rarezas) y fueron reacios a cambiar a otras tecnologías LAN cuando entraron en escena. En segundo lugar, Token Ring, FDDI y ATM eran más complejas y caras que Ethernet, lo que desanimó a los administradores de redes a cambiar. En tercer lugar, la razón más determinante para cambiar a otra tecnología LAN (como FDDI o ATM) era normalmente la más alta velocidad de datos de la nueva tecnología; sin embargo, Ethernet siempre se defendió produciendo versiones que operaban a velocidades iguales o incluso mayores. Ethernet conmutada se introdujo también a principios de la década de 1990, lo que aumentó sus tasas de datos efectivas. Por último, dado que Ethernet ha sido tan popular, el hardware Ethernet (en concreto, los adaptadores y los conmutadores) ha llegado a ser cómodo y notablemente barato.

La LAN Ethernet original fue inventada a mediados de la década de 1970 por Bob Metcalfe y David Boggs. La Figura 5.20 muestra el esquema que realizó Metcalfe. En la figura puede observar que la LAN Ethernet original utilizaba un bus coaxial para interconectar los nodos. Las topologías de bus para Ethernet realmente se mantuvieron durante la década de 1980 y hasta mediados de la década de 1990. Ethernet con una topología de bus es una LAN de difusión (todas las tramas transmitidas viajan hasta *todos* los adaptadores conectados al bus y son procesadas en ellos).

A finales de la década de 1990, la mayor parte de las empresas y universidades habían reemplazado sus redes LAN por instalaciones Ethernet utilizando topologías en estrella basadas en concentradores (*hubs*). Como se muestra en la Figura 5.21, en tal instalación los hosts (y los routers) están directamente conectados a un concentrador mediante un cable de

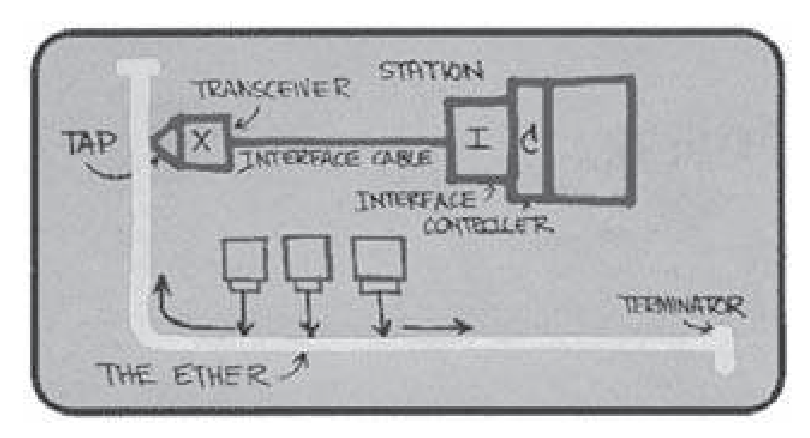


Figura 5.20 • El diseño original de Metcalfe llevó al estándar Ethernet 10BASE5, que incluía un cable de interfaz que conectaba la tarjeta adaptadora Ethernet a un transductor externo.

cobre de par trenzado. Un **concentrador** es un dispositivo de la capa física que actúa sobre los bits individuales en lugar de sobre las tramas. Cuando un bit, que representa un cero o un uno, llega procedente de una interfaz, el concentrador simplemente vuelve a crear el bit, incrementa su intensidad de energía y lo transmite a todas las demás interfaces. Por tanto, Ethernet con una topología de estrella basada en concentrador es también una red LAN de difusión (cuando un concentrador recibe un bit en una de sus interfaces, envía una copia al resto de sus interfaces). En particular, si un concentrador recibe tramas procedentes de dos interfaces distintas al mismo tiempo, se produce una colisión y los nodos que crean las tramas tendrán que retransmitirlas.

A principios de la década de 2000 Ethernet experimentó una cambio evolutivo aún mayor. Las instalaciones Ethernet continuaron utilizando una topología en estrella, pero el concentrador central fue reemplazado por un **conmutador** (*switch*). Examinaremos en pro-

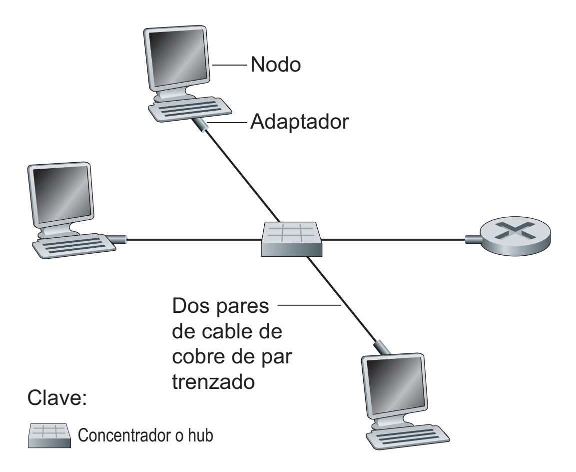


Figura 5.21 • Topología en estrella para Ethernet: los nodos están interconectados mediante un concentrador.

fundidad las redes Ethernet conmutadas más adelante en este capítulo. Por el momento, sólo mencionaremos que un conmutador no es sólo un dispositivo sin colisiones, sino que también lleva a cabo la conmutación de paquetes mediante un almacenamiento y reenvío de buena fe; pero, a diferencia de los routers, que operan hasta la capa 3, un conmutador opera sólo hasta la capa 2.

5.5.1 Estructura de la trama de Ethernet

Podemos aprender mucho acerca de Ethernet examinando la trama que utiliza, la cual se muestra en la Figura 5.22. Con el fin de proporcionar a esta exposición acerca de las tramas de Ethernet un contexto tangible, consideremos el envío de un datagrama IP desde un host a otro host, estando ambos hosts en la misma red LAN Ethernet (por ejemplo, en la LAN Ethernet de la Figura 5.21). (Aunque la carga útil de nuestra trama Ethernet es un datagrama IP, tenga en cuenta que una trama Ethernet puede transportar también otros paquetes de la capa de red.) El adaptador del emisor, adaptador A, tiene la dirección MAC AA-AA-AA-AA-AA-AA y el adaptador del receptor, adaptador B, tiene la dirección MAC BB-BB-BB-BB-BB-BB. El adaptador del emisor encapsula el datagrama IP dentro de una trama Ethernet y pasa dicha trama a la capa física. El adaptador del receptor recibe la trama de la capa física, extrae el datagrama IP y lo pasa a la capa de red. En este contexto, examinemos los seis campos de la trama Ethernet mostrada en la Figura 5.22.

- *Campo de datos (46 a 1.500 bytes).* Este campo transporta el datagrama IP. La unidad máxima de transmisión (MTU) de Ethernet es 1.500 bytes, lo que quiere decir que si el datagrama IP excede de 1.500 bytes, entonces el host tiene que fragmentar el datagrama, como se ha explicado en la Sección 4.4.1. El tamaño mínimo del campo de datos es 46 bytes, lo que significa que si el datagrama IP tiene menos de 46 bytes, el campo de datos tiene ser rellenado hasta los 46 bytes. Cuando se utiliza el rellenado, los datos pasados a la capa de red contienen tanto el relleno como el datagrama IP. La capa de red utiliza el campo longitud de la cabecera del datagrama IP para eliminar el relleno.
- *Dirección de destino (6 bytes).* Este campo contiene la dirección MAC del adaptador de destino, BB-BB-BB-BB-BB-BB. Cuando el adaptador B recibe una trama Ethernet cuya dirección de destino es BB-BB-BB-BB-BB-BB o la dirección MAC de difusión, pasa el contenido del campo de datos de la trama a la capa de red; si recibe una trama con cualquier otra dirección MAC, descarta la trama.
- *Dirección de origen (6 bytes).* Este campo contiene la dirección MAC del adaptador que transmite la trama hacia la LAN; en este ejemplo, AA-AA-AA-AA-AA-AA.
- *Campo de tipo (2 bytes).* El campo de tipo permite a Ethernet multiplexar los protocolos de la capa de red. Para comprender esto, tenemos que tener en cuenta que los hosts pueden utilizar otros protocolos de la capa de red además de IP. De hecho, un determinado host puede dar soporte a múltiples protocolos de la capa de red utilizando protocolos distintos para las diferentes aplicaciones. Por esta razón, cuando llega la trama Ethernet al adaptador B, éste necesita saber a qué protocolo de la capa de red debe pasar (es decir, demultiplexar) el contenido del campo de datos. IP y otros protocolos de la capa red (por ejemplo, Novell IPX o AppleTalk) tienen su propio número de tipo estandarizado. Además, el protocolo ARP (estudiado en la sección anterior) tiene su propio número de tipo y si la trama que llega contiene un paquete ARP (es decir, el campo de tipo contiene el hexadecimal 0806), el paquete ARP será demultiplexado y entregado al protocolo ARP.

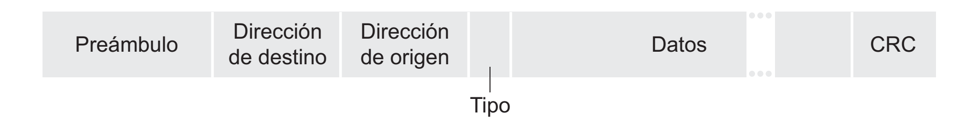


Figura 5.22 • Estructura de la trama Ethernet.

Observe que el campo de tipo es análogo al campo de protocolo del datagrama de la capa de red y a los campos de número de puerto del segmento de la capa de transporte; todos estos campos sirven para enlazar un protocolo de una capa con un protocolo de la capa superior.

- *Comprobación de redundancia cíclica (CRC) (4 bytes).* Como se ha visto en la Sección 5.2.3, el propósito del campo CRC es permitir que el adaptador del receptor, el adaptador B, detecte los errores de bit de la trama.
- *Preámbulo (8 bytes).* La trama Ethernet comienza con el campo preámbulo de 8 bytes. Cada uno de los siete primeros bytes tiene el valor 10101010 y el último byte tiene el valor 10101011. Los siete primeros bytes sirven para "despertar" a los adaptadores de recepción y sincronizar sus relojes con el reloj del emisor. ¿Por qué podrían estar desincronizados los relojes? Tenga en cuenta que el objetivo del adaptador A es transmitir la trama a 10 Mbps, 100 Mbps o 1 Gbps, dependiendo del tipo de LAN Ethernet. Sin embargo, dado que nada es absolutamente perfecto, el adaptador A no transmitirá la trama a una velocidad exactamente igual a la objetivo; siempre existe cierta *deriva* respecto de dicha velocidad, una deriva que no es conocida *a priori* por los restantes adaptadores de la LAN. Un adaptador de recepción puede sincronizarse con el reloj del adaptador A sincronizándose simplemente con los bits de los siete primeros bytes del preámbulo. Los últimos 2 bits del octavo byte del preámbulo (los dos primeros 1s consecutivos) alertan al adaptador B de que va a llegar "información importante".

Ethernet utiliza la transmisión en banda base; es decir, el adaptador envía una señal digital directamente al canal de difusión. La tarjeta de interfaz no desplaza la señal a otra banda de frecuencias, como ocurre con el sistema ADSL y los sistemas de módem por cable. Muchas tecnologías Ethernet (como por ejemplo, 10BASE-T) también utilizan la codificación Manchester, como se muestra en la Figura 5.23. Con la codificación Manchester, cada bit contiene una transición; un 1 indica una transición del nivel alto al nivel bajo, mientras que un 0 indica una transición del nivel bajo al nivel alto. La razón de utilizar la codificación Manchester es que los relojes de los adaptadores del emisor y del receptor no están perfectamente sincronizados. Al incluir una transición a mitad de cada bit, el host receptor puede sincronizar su reloj con el del host emisor. Una vez que el reloj del adaptador del receptor está sincronizado, el receptor puede delinear cada bit y determinar si se trata de un 1 o de un 0. La codificación Manchester es una operación de la capa física más que de la capa de enlace; no obstante, es necesario que expliquemos aquí brevemente porque se utiliza exhaustivamente en Ethernet.

Servicio sin conexión no fiable

Todas las tecnologías Ethernet proporcionan un servicio sin conexión a la capa de red; es decir, cuando el adaptador A desea enviar un datagrama al adaptador B, el adaptador A encapsula el datagrama en una trama Ethernet y la envía a la red LAN, sin establecer previamente

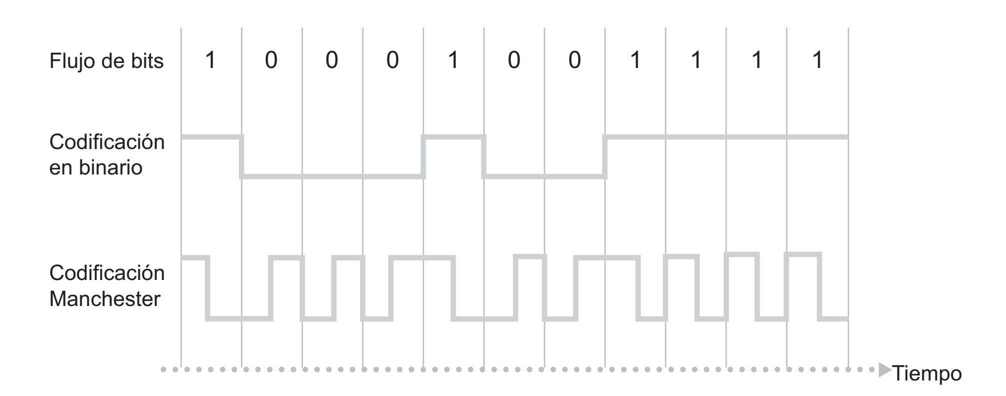


Figura 5.23 • Codificación Manchester.

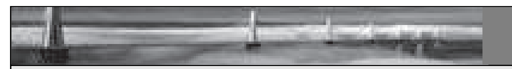
un acuerdo con el adaptador B. Este servicio sin conexión de la capa 2 es análogo al servicio de datagramas de la capa 3 de IP y al servicio sin conexión de la capa 4 de UDP.

Las tecnologías Ethernet proporcionan un servicio no fiable a la capa de red. Específicamente, cuando el adaptador B recibe una trama procedente del adaptador A ejecuta una comprobación CRC de la trama, pero ni envía un mensaje de reconocimiento cuando la trama pasa la comprobación CRC, ni envía un mensaje de reconocimiento negativo cuando la comprobación CRC falla. Cuando una trama no pasa la comprobación CRC, el adaptador B simplemente la descarta. Por tanto, el adaptador A no sabe si la trama que ha transmitido ha llegado al adaptador B y ha superado la comprobación CRC. Esta ausencia de un transporte fiable (en la capa de enlace) hace que Ethernet sea una tecnología simple y barata. Pero también significa que el flujo de datagramas pasado a la capa de red puede presentar "huecos".

Si existen huecos porque se han descartado tramas Ethernet, ¿la aplicación del host B ve también esos huecos? Como hemos visto en el Capítulo 3, esto dependerá de si la aplicación está utilizando UDP o TCP. Si la aplicación está empleando UDP, entonces la aplicación del host B verá los huecos en los datos. Por el contrario, si la aplicación está utilizando TCP, entonces TCP en el host B no confirmará los datos contenidos en las tramas descartadas, obligando a TCP en el host A a realizar retransmisiones. Observe que cuando TCP retransmite los datos, éstos finalmente volverán al adaptador Ethernet en el que fueron descartados. Por tanto, en este sentido, Ethernet retransmite los datos, aunque no es consciente de si está transmitiendo un nuevo datagrama con nuevos datos, o un datagrama que contiene datos que ya han sido transmitidos al menos una vez.

5.5.2 CSMA/CD: protocolo de acceso múltiple de Ethernet

Cuando los nodos están interconectados mediante un concentrador (en oposición a un conmutador de la capa de enlace), como se muestra en la Figura 5.21, la LAN Ethernet es una auténtica LAN de difusión; es decir, cuando un adaptador transmite una trama, todos los adaptadores de la LAN reciben esa trama. Dado que Ethernet puede emplear la comunicación por difusión, necesita un protocolo de acceso múltiple. Ethernet utiliza el famoso protocolo de acceso múltiple CSMA/CD. Recuerde que, como hemos visto en la Sección 5.3, CSMA/CD hace lo siguiente:



HISTORIA

BOB METCALFE Y ETHERNET

Como estudiante de doctorado en la Universidad de Harvard a principios de la década de 1970, Bob Metcalfe trabajaba en la red ARPAnet en el MIT. Durante sus estudios, también conoció el trabajo de Abramson con ALOHA y los protocolos de acceso aleatorio. Después de terminar su doctorado y justo antes de comenzar a trabajar en el Centro de investigación de Xerox Palo Alto (Xerox PARC), estuvo durante tres meses con Abramson y sus colegas de la universidad de Hawaii y pudo obtener información de primera mano sobre ALOHAnet. En Xerox PARC, Metcalfe también conoció las computadoras Alto, que en muchos sentidos fueron las precursoras de las computadoras personales de la década de 1980. Metcalfe vió la necesidad de conectar en red estas computadoras mediante algún sistema económico. Así, armado con sus conocimientos sobre ARPAnet, ALOHAnet y los protocolos de acceso aleatorio, Metcalfe (junto con su colega David Boggs) inventó Ethernet.

La Ethernet original de Metcalfe y Boggs operaba a 2,94 Mbps y podía conectar hasta 256 hosts separados hasta aproximadamente un kilómetro y medio. Metcalfe y Boggs tuvieron éxito al conseguir que la mayoría de los investigadores de Xerox PARC pudieran comunicarse a través de sus computadoras Alto. Metcalfe forjó después una alianza entre Xerox, Digital e Intel para establecer Ethernet a 10 Mbps como estándar del IEEE. Xerox no mostró demasiado interés en la comercialización de Ethernet. En 1979, Metcalfe montó su propia empresa, 3Com, que desarrolló y comercializó tecnología de redes, incluyendo la tecnología Ethernet. En concreto, 3Com desarrolló y puso en el mercado tarjetas Ethernet a principios de la década de 1980 para las tremedamente populares computadoras personales de IBM. Metcalfe abandonó 3Com en 1990, cuando tenía 2.000 empleados y unos ingresos de 400 millones de dólares.

1. Un adaptador puede comenzar a transmitir en cualquier instante; es decir, no existe el concepto de partición de tiempo.
2. Un adaptador nunca transmite una trama cuando detecta que algún otro adaptador está transmitiendo; es decir, utiliza un mecanismo de sondeo de portadora.
3. Un adaptador que está transmitiendo aborta su transmisión tan pronto como detecta que otro adaptador también está transmitiendo; es decir, utiliza un mecanismo de detección de colisiones.
4. Antes de intentar llevar a cabo una retransmisión, un adaptador espera un intervalo de tiempo aleatorio que normalmente es más pequeño que el tiempo que se tarda en transmitir una trama.

Estos mecanismos proporcionan a CSMA/CD un rendimiento mucho mejor que el del protocolo ALOHA con particiones en un entorno LAN. De hecho, si el retardo máximo de propagación entre estaciones es muy pequeño, la eficiencia de CSMA/CD puede aproximarse al 100 por ciento. Observe también que el segundo y tercer mecanismos de la lista anterior requieren que los adaptadores de Ethernet sean capaces de (1) detectar cuándo algún otro adaptador está transmitiendo y (2) detectar una colisión mientras están transmitiendo. Los adaptadores Ethernet realizan estas dos tareas midiendo los niveles de tensión antes y durante las transmisiones.

Los adaptadores ejecutan el protocolo CSMA/CD sin coordinación explícita con los demás adaptadores existentes en la red Ethernet. Dentro de un adaptador específico, el protocolo CSMA/CD opera de la siguiente forma:

1. El adaptador obtiene un datagrama de la capa de red, prepara una trama Ethernet y la coloca en un buffer del adaptador.
2. Si el adaptador detecta que el canal está inactivo (es decir, durante 96 periodos de bit el adaptador no recibe intensidad de señal procedente del canal), comienza a transmitir la trama. Si el adaptador detecta que el canal está ocupado, espera hasta comprobar que no hay intensidad de señal (más otros 96 periodos de bit) y luego comienza a transmitir la trama.
3. Mientras está transmitiendo, el adaptador monitoriza la presencia de señales procedentes de otros adaptadores. Si el adaptador transmite la trama completa sin detectar ninguna señal procedente de otros adaptadores, concluye que ha terminado su trabajo con esa trama.
4. Si el adaptador detecta intensidad de señal procedente de otros adaptadores mientras está transmitiendo, deja de transmitir su trama y transmite una señal de interferencia (*jam*) de 48 bits.
5. Después de abortar la transmisión de la trama (es decir, de transmitir la señal de interferencia), el adaptador entra en la fase de espera exponencial (***backoff* exponencial**). Específicamente, a la hora de transmitir una determinada trama, después de experimentar la *n-ésima* colisión para esa trama, el adaptador selecciona un valor aleatorio para K del conjunto $\{0,1,2, \ldots, 2^m - 1\}$, donde $m = \min(n,10)$. El adaptador espera entonces K 512 periodos de bit y vuelve al Paso 2.

Es necesario que comentemos algunas cosas acerca del protocolo CSMA/CD. El propósito de la señal de interferencia es el de garantizar que todos los adaptadores que están transmitiendo sean conscientes de la colisión. Veamos un ejemplo. Suponga que el adaptador A comienza a transmitir una trama y justo antes de que la señal de A llegue al adaptador B éste empieza a transmitir. Por tanto, B sólo transmitirá unos pocos bits antes de abortar su transmisión. Esos pocos bits se propagarán hasta A, pero es posible que no generen una intensidad de señal suficiente como para que A detecte la colisión. Para asegurarse de que A detecta la colisión (de modo que también pueda abortar su transmisión), B transmite la señal de interferencia de 48 bits.

A continuación vamos a ver el algoritmo de *backoff* exponencial. Lo primero que tenemos que observar aquí es que un periodo de bit (es decir, el tiempo que se tarda en transmitir un único bit) es muy corto; en una red Ethernet a 10 Mbps, un periodo de bit es igual a 0,1 microsegundos. Veamos un ejemplo. Suponga que un adaptador intenta transmitir una trama por primera vez y mientras está transmitiendo detecta una colisión. El adaptador selecciona entonces $K = 0$ con una probabilidad de 0,5 o $K = 1$ con una probabilidad de 0,5. Si el adaptador elige $K = 0$, entonces pasa de forma inmediata al Paso 2 después de transmitir la señal de interferencia. Si el adaptador elige $K = 1$, entonces espera 51,2 microsegundos antes de volver al Paso 2. Después de una segunda colisión, K se selecciona con la misma probabilidad del conjunto $\{0, 1, 2, 3\}$. Después de tres colisiones, K se elige con la misma probabilidad del conjunto $\{0, 1, 2, 3, 4, 5, 6, 7\}$. Después de 10 o más colisiones, K se elige con la misma probabilidad del conjunto $\{0, 1, 2, \ldots, 1023\}$. Por tanto, el tamaño de los conjuntos de los que se selecciona K crece exponencialmente con el número de colisio-

nes (hasta $n = 10$); ésta es la razón por la que el algoritmo de *backoff* de Ethernet se denomina *backoff* exponencial.

El estándar Ethernet impone límites a la distancia existente entre dos nodos cualesquiera. Estos límites garantizan que si el adaptador A elige un valor más pequeño de K que todos los demás adaptadores implicados en una colisión, entonces el adaptador A podrá transmitir su trama sin experimentar una nueva colisión. En los problemas de repaso veremos más detalladamente esta propiedad.

¿Por qué se utiliza el algoritmo de *backoff* exponencial? ¿Por qué no seleccionar K, por ejemplo, del conjunto {0, 1, 2, 3, 4, 5, 6, 7} después de cada colisión? La razón es que cuando un adaptador experimenta su primera colisión, no tiene ni idea de cuántos adaptadores están implicados en la misma. Si el número de adaptadores que han colisionado es pequeño, tiene sentido seleccionar K de un conjunto pequeño de valores bajos. Por el contrario, si son muchos los adaptadores implicados en la colisión, tiene sentido elegir K de un conjunto más grande y más disperso de valores (¿por qué?). Aumentando el tamaño del conjunto después de cada colisión, el adaptador se adapta apropiadamente a estos distintos escenarios.

Observe también que cada vez que un adaptador prepara una nueva trama para su transmisión ejecuta el algoritmo CSMA/CD presentado anteriormente, no teniendo en cuenta las colisiones que hayan tenido lugar en el pasado reciente. Por tanto, es posible que un adaptador con una nueva trama sea capaz de llevar a cabo una transmisión con éxito de forma inmediata mientras que otros adaptadores se encuentran en el estado de *backoff* exponencial.

Eficiencia de Ethernet

Cuando sólo un nodo tiene una trama para enviar, el nodo puede transmitir a la velocidad máxima de la tecnología Ethernet (por ejemplo, 10 Mbps, 100 Mbps o 1 Gbps). Sin embargo, si muchos nodos tienen tramas que transmitir, la velocidad efectiva de transmisión del canal puede ser mucho menor. Definimos eficiencia de Ethernet como la fracción (a largo plazo) de tiempo durante el que las tramas están siendo transmitidas al canal sin colisiones, cuando existe un gran número de nodos activos, teniendo cada uno de ellos una gran cantidad de tramas para enviar. Para obtener una buena aproximación de la eficiencia de Ethernet, definimos d_{prop} como el tiempo máximo que tarda la intensidad de la señal en propagarse entre dos adaptadores. Sea d_{trans} el tiempo necesario para transmitir una trama Ethernet de tamaño máximo (aproximadamente 1,2 milisegundos para Ethernet a 10 Mbps). Una demostración de la expresión de la eficiencia de Ethernet queda fuera del alcance de este libro (consulte [Lam 1980] y [Bertsekas 1991]). A continuación proporcionamos simplemente la siguiente aproximación:

$$\text{Eficiencia} = \frac{1}{1 + 5d_{prop}/\, d_{trans}}$$

A partir de esta fórmula vemos que cuando d_{prop} se aproxima a 0, la eficiencia tiende a 1. Esto confirma la idea intuitiva de que si el retardo de propagación es cero, los nodos que han colisionado abortarán de forma inmediata sus transmisiones, evitando así que el canal se desperdicie. También, cuando d_{trans} se hace muy grande, la eficiencia tiende a 1. Esto es igualmente intuitivo, ya que cuando una trama se apropia del canal se mantendrá en él

durante bastante tiempo y, por tanto, el canal estará realizando un trabajo productivo la mayor parte del tiempo.

5.5.3 Tecnologías Ethernet

En nuestra exposición anterior hemos hecho a referencia a Ethernet como si fuera el único estándar de protocolo. Pero, en realidad, existen *muchas* versiones diferentes de Ethernet, algunas con acrónimos algo complicados como 10BASE-T, 10BASE-2, 100BASE-T, 1000BASE-LX y 10GBASE-T. Éstas y otras muchas tecnologías Ethernet han sido estandarizadas a lo largo de los años por el grupo de trabajo IEEE 802.3 CSMA/CD (Ethernet) [IEEE 802.3 2009]. Aunque estos acrónimos pueden parecer algo complicados, realmente tienen su lógica. La primera parte del acrónimo hace referencia a la velocidad del estándar: 10, 100, 1000 o 10G, para 10 Megabits (por segundo), 100 Megabits, Gigabit y 10 Gigabits Ethernet, respectivamente. "BASE" hace referencia a la tecnología Ethernet banda base, que significa que el medio físico sólo transporta tráfico Ethernet; casi todos los estándares 802.3 son para Ethernet banda base. La parte final del acrónimo se refiere al medio físico en sí. Ethernet es una especificación tanto de la capa de enlace como de la capa física y puede emplear diversos medios físicos entre los que se incluyen el cable coaxial, el cable de cobre y la fibra. Generalmente, la "T" hace referencia al cable de cobre de par trenzado.

Históricamente, Ethernet fue concebida inicialmente como un segmento de cable coaxial, como se muestra en la Figura 5.20. Los primeros estándares 10BASE-2 y 10BASE5 especificaban Ethernet a 10 Mbps sobre dos tipos de cable coaxial, cada uno de ellos limitado a una longitud de 500 metros. Podían obtenerse recorridos más largos utilizando un **repetidor** (un dispositivo de la capa física que recibe una señal en su entrada y regenera la señal en la salida). Un cable coaxial, como el mostrado en la Figura 5.20, se corresponde con nuestra visión de Ethernet como un medio de difusión (todas las tramas transmitidas por una interfaz son recibidas en otras interfaces), y el protocolo CDMA/CD de Ethernet resuelve muy bien el problema del acceso múltiple. Los nodos simplemente se conectan al cable, y *voila*, tenemos una red de área local.

La tecnología Ethernet ha pasado por una serie de pasos evolutivos a lo largo de los años y la Ethernet actual es muy diferente de los diseños de las topologías de bus originales que utilizaban cable coaxial. En la mayor parte de las instalaciones actuales, los nodos se conectan a un conmutador mediante segmentos punto a punto hechos de cable de cobre de par trenzado o de cable de fibra óptica, como se muestra en la Figura 5.24.

A mediados de la década de 1990 Ethernet se estandarizó a 100 Mbps, tecnología 10 veces más rápida que la Ethernet a 10 Mbps. El protocolo MAC Ethernet original y el formato de trama se conservaron, pero se definieron capas físicas de mayor velocidad para cable de cobre (100BASE-T) y fibra (100BASE-FX, 100BASE-SX, 100BASE-BX). La Figura 5.25 muestra estos distintos estándares, junto con el protocolo MAC Ethernet y el formato de trama comunes. La tecnología Ethernet a 100 Mbps está limitada a una distancia de 100 metros sobre cable de par trenzado y a varios kilómetros para cable de fibra, lo que permite conectar conmutadores Ethernet situados en diferentes edificios.

Ethernet Gigabit es una extensión de los estándares Ethernet a 10 Mbps y 100 Mbps de mayor éxito. Ofreciendo una tasa de datos en bruto de 1.000 Mbps, Ethernet Gigabit mantiene una compatibilidad total con la enorme base instalada de equipos Ethernet. El estándar para Ethernet Gigabit, IEEE 802.3z, hace lo siguiente:

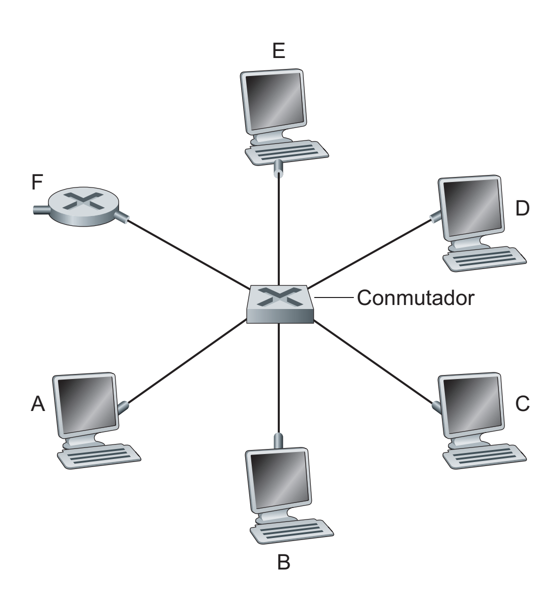


Figura 5.24 • Conmutador de la capa de enlace que interconecta seis nodos.

- Utiliza el formato de trama Ethernet estándar (Figura 5.22) y es compatible hacia atrás con las tecnologías 10BASE-T y 100BASE-T. Esto permite una fácil integración de Ethernet Gigabit con la base instalada existente de equipos Ethernet.
- Permite el uso de enlaces punto a punto, así como de canales de difusión compartidos. Los enlaces punto a punto utilizan conmutadores mientras que los canales de difusión emplean concentradores, como se ha descrito anteriormente. En la jerga de Ethernet Gigabit, los concentradores reciben el nombre de *distribuidores con buffer*.
- Utiliza CSMA/CD para los canales de difusión compartidos. Para obtener una eficiencia aceptable, la distancia máxima entre nodos tiene que ser restringida de forma estricta.
- Permite la operación *full-duplex* a 1.000 Mbps en ambas direcciones en los canales punto a punto.

Operando inicialmente sobre cable de fibra óptica, ahora Ethernet Gigabit es capaz de operar sobre cable UTP de categoría 5. En 2007 se estandarizó Ethernet a 10 Gbps (10GBASE-T), proporcionando aún mayores capacidades a las redes LAN Ethernet.

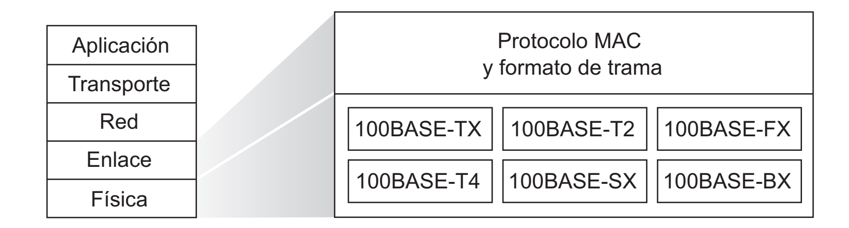


Figura 5.25 • Estándares Ethernet a 100 Mbps: una capa de enlace común y diferentes capas físicas.

Vamos a concluir nuestra exposición sobre la tecnología Ethernet planteando una cuestión que puede que le haya estado rondando la cabeza. En la época de las topologías de bus y de las topologías en estrella basadas en concentrador, Ethernet era claramente un enlace de difusión (como se ha definido en la Sección 5.3) en el que se producían colisiones entre tramas cuando los nodos transmitían al mismo tiempo. Para tratar estas colisiones, el estándar Ethernet incluyó el protocolo CSMA/CD, que es particularmente efectivo en las redes LAN cableadas de difusión con un radio geográfico pequeño. Pero, si el uso prevalente de hoy día es una topología en estrella basada en conmutadores, que utiliza la conmutación de paquetes de almacenamiento y reenvío, ¿existe realmente la necesidad de un protocolo MAC Ethernet? Como veremos en la Sección 5.6, un conmutador coordina sus transmisiones y nunca reenvía más de una trama a la misma interfaz en un determinado instante. Además, los conmutadores modernos son *full-duplex*, por lo que un conmutador y un nodo pueden enviarse tramas entre sí simultáneamente sin interferir. En otras palabras, en una red LAN Ethernet basada en conmutadores no se producen colisiones y, por tanto, no se necesita un protocolo MAC.

Como hemos visto, las tecnologías Ethernet actuales son *muy* diferentes de la tecnología Ethernet original concebida por Metcalfe y Boggs hace más de 30 años; las velocidades se han incrementado en tres órdenes de magnitud, las tramas Ethernet son transportadas por una amplia variedad de medios, la tecnología Ethernet conmutada se ha convertido en la tecnología dominante y ahora incluso el protocolo MAC con frecuencia no es necesario. ¿Sigue siendo todo esto *realmente* Ethernet? Por supuesto, la respuesta es "sí, por definición". Sin embargo, es interesante observar que a pesar de todos estos cambios, hay algo que se ha mantenido inalterado a lo largo de estos 30 años: el formato de la trama Ethernet. Quizá sea entonces ésta la única verdadera pieza importante del estándar Ethernet.

5.6 Conmutadores de la capa de enlace

Como se muestra en la Figura 5.26, las redes LAN Ethernet modernas utilizan una topología en estrella, estando cada nodo conectado a un conmutador central. Hasta el momento no hemos entrado en detalles sobre lo que realmente hace un conmutador y cómo funciona. La función de un conmutador es recibir las tramas de la capa de enlace entrantes y reenviarlas a los enlaces de salida; enseguida vamos a estudiar la función de reenvío en detalle. El propio conmutador es **transparente** para los nodos; es decir, un nodo dirige una trama a otro nodo (en lugar de dirigirla al conmutador) y la envía a la red LAN, sin ser consciente de que un conmutador recibirá la trama y la reenviará a los demás nodos. La velocidad a la que llegan las tramas a cualquiera de las interfaces de salida del conmutador puede ser temporalmente mayor que la capacidad del enlace de dicha interfaz. Para enfrentarse a este problema, las interfaces de salida del conmutador disponen de buffers, de forma muy parecida a como las interfaces de salida de un router disponen de buffers para los datagramas. Veamos ahora detenidamente cómo funciona un conmutador.

5.6.1 Reenvío y filtrado

El **filtrado** es la función del conmutador que determina si una trama debe ser reenviada a alguna interfaz o debe ser descartada. El **reenvío** es la función del conmutador que determina las interfaces a las que un trama debe dirigirse y luego envía la trama a esas interfaces.

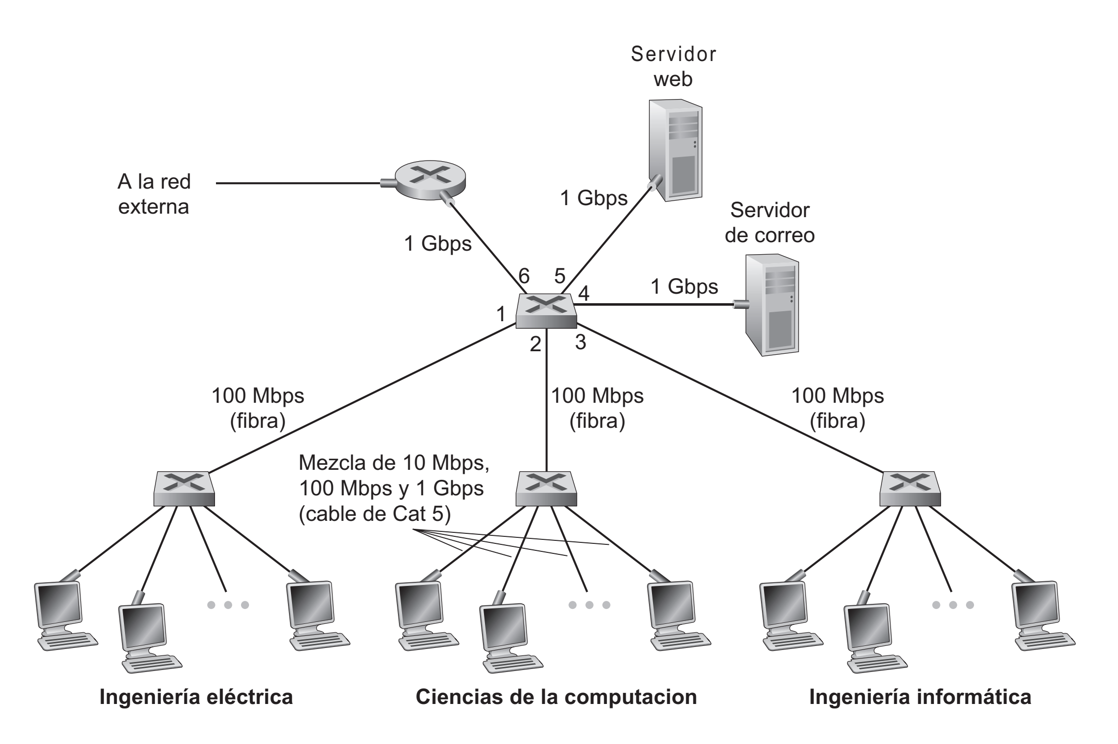


Figura 5.26 • Una red institucional que utiliza una combinación de concentradores, conmutadores Ethernet y un router.

Las funciones de filtrado y reenvío del conmutador se realizan utilizando la **tabla del conmutador**. Esta tabla contiene entradas para algunos nodos, no necesariamente todos, de una red LAN. Una entrada de la tabla del conmutador contiene (1) la dirección MAC de un nodo, (2) la interfaz del conmutador que lleva hacia el nodo y (3) el instante en el que la entrada para el nodo fue incluida en la tabla. Un ejemplo de tabla de conmutación para el conmutador superior de la Figura 5.26 se muestra en la Figura 5.27. Aunque esta descripción de reenvío de tramas puede parecer muy similar a lo que hemos visto sobre el reenvío de datagramas en el Capítulo 4, veremos que existen diferencias importantes. Una de estas diferencias es que los conmutadores reenvían los paquetes basándose en las direcciones MAC en lugar de en la direcciones IP. También veremos que la tabla del conmutador se construye de forma muy distinta a como se crea la tabla de reenvío de un router.

Para comprender cómo funciona el filtrado y el reenvío en un conmutador, suponga que una trama con la dirección de destino DD-DD-DD-DD-DD-DD llega a la interfaz *x* del

Dirección	Interfaz	Hora
62-FE-F7-11-89-A3	1	9:32
7C-BA-B2-B4-91-10	3	9:36
....	....	....

Figura 5.27 • Parte de la tabla del conmutador superior de la Figura 5.26.

conmutador. Éste buscará en su tabla la dirección MAC DD-DD-DD-DD-DD-DD. Se plantean tres posibilidades:

- No hay ninguna entrada en la tabla para DD-DD-DD-DD-DD-DD. En este caso, el conmutador reenvía copias de la trama a los buffers de salida que preceden a *todas* las interfaces salvo a la interfaz *x*. En otras palabras, si no hay ninguna entrada para la dirección de destino, el conmutador difunde la trama.
- Existe una entrada en la tabla que asocia DD-DD-DD-DD-DD-DD con la interfaz *x*. En este caso, la trama procede de un segmento de la LAN que contiene al adaptador DD-DD-DD-DD-DD-DD. No existe la necesidad de reenviar la trama a ninguna de las restantes interfaces, el conmutador lleva a cabo la función de filtrado descartando la trama.
- Existe una entrada en la tabla que asocia DD-DD-DD-DD-DD-DD con la interfaz *y* *x*. En este caso, la trama tiene que ser reenviada al segmento de la LAN conectado a la interfaz *y*. El conmutador lleva a cabo su función de reenvío colocando la trama en un buffer de salida que precede a la interfaz *y*.

Apliquemos estas reglas al conmutador de la parte superior de la Figura 5.26 y a su tabla mostrada en la Figura 5.27. Suponga que una trama con la dirección de destino 62-FE-F7-11-89-A3 llega al conmutador desde la interfaz 1. El conmutador examina su tabla y ve que el destino está en el segmento de LAN conectado a la interfaz 1 (es decir, Ingeniería Eléctrica). Esto significa que la trama ya ha sido difundida en el segmento de LAN que contiene el destino. Por tanto, el conmutador filtra (es decir, descarta) la trama. Suponga ahora que una trama con la misma dirección de destino llega de la interfaz 2. De nuevo, el conmutador examina su tabla y ve que el destino está en la dirección de interfaz 1; por tanto, reenvía la trama al buffer de salida que precede a la interfaz 1. Con este ejemplo debería quedar claro que a medida que la tabla del conmutador se completa y precisa, el conmutador reenvía las tramas hacia los destinos sin llevar a cabo ninguna difusión.

En este sentido, un conmutador es "más inteligente" que un concentrador. Pero, ¿como se configura la tabla del conmutador? ¿Existen equivalentes para la capa de enlace de los protocolos de enrutamiento de la capa de red? ¿O tiene un administrador sobrecargado de trabajo que configurar manualmente la tabla del conmutador?

5.6.2 Auto-aprendizaje

Los conmutadores tienen la fantástica propiedad (especialmente para los administradores de redes sobrecargados de trabajo) de que su tabla se construye de forma automática, dinámica y autónoma, sin intervención de un administrador de redes ni de ningún protocolo de configuración. En otras palabras, los conmutadores son **auto-aprendices**. Esta capacidad se lleva cabo de la forma siguiente:

1. Inicialmente, la tabla del conmutador está vacía.
2. Para cada trama entrante recibida en una interfaz, el conmutador almacena en su tabla (1) la dirección MAC especificada en el *campo dirección de origen* de la trama, (2) la interfaz de la que procede la trama y (3) la hora actual. De esta forma, el conmutador registra en su tabla el segmento de la LAN en la que reside el nodo emisor. Si todos los nodos de la LAN terminan enviando una trama, entonces todos los nodos terminarán estando registrados en la tabla.

Dirección	Interfaz	Hora
01-12-23-34-45-56	2	9:39
62-FE-F7-11-89-A3	1	9:32
7C-BA-B2-B4-91-10	3	9:36
....	....	....

Figura 5.28 • El conmutador aprende la ubicación de un adaptador con la dirección 01-12-23-34-45-56.

3. El conmutador borra una dirección de la tabla si no se recibe ninguna trama con esa dirección como dirección de origen transcurrido un cierto periodo de tiempo (el **tiempo de envejecimiento**). De esta forma, si un PC es sustituido por otro (con un adaptador diferente), la dirección MAC del PC original será eliminado de la tabla del conmutador.

Examinemos la propiedad de auto-aprendizaje del conmutador superior de la Figura 5.26 y su tabla de conmutación correspondiente, mostrada en la Figura 5.27. Suponga que a las 9:39 una trama con la dirección de origen 01-12-23-34-45-56 llega procedente de la interfaz 2. Suponga también que esa dirección no está incluida en la tabla del conmutador. Entonces el conmutador añade una nueva entrada, como se muestra en la Figura 5.28.

Continuando con el ejemplo, suponga que el tiempo de envejecimiento para este conmutador es de 60 minutos, y que entre las 9:32 y las 10:32 no le llega ninguna trama con la dirección de origen 62-FE-F7-11-89-A3. Entonces, a las 10:32, el conmutador eliminará esta dirección de su tabla.

Los conmutadores son **dispositivos plug-and-play** porque no requieren intervención ni de un administrador de redes ni de los usuarios. Un administrador de redes que desee instalar un conmutador no tiene que hacer nada más que conectar los segmentos de la LAN a las interfaces del conmutador. El administrador no tiene que configurar las tablas del conmutador en el momento de la instalación ni cuando se elimina un host de uno de los segmentos de la LAN. Los conmutadores también permiten la comunicación *full-duplex*, lo que significa que para cualquier enlace que conecte un nodo con el conmutador, tanto el nodo como el conmutador pueden transmitir al mismo tiempo sin que se produzcan colisiones.

5.6.3 Propiedades de la conmutación de la capa de enlace

Una vez que hemos descrito el funcionamiento básico de un conmutador de la capa de enlace, vamos a pasar a ver sus características y propiedades. Utilizando la red LAN de la Figura 5.24 podemos identificar varias ventajas de utilizar conmutadores, en lugar de enlaces de difusión como las topologías de bus o basadas en concentradores.

- *Eliminación de las colisiones*. En una red LAN construida con conmutadores (y sin concentradores) no se desperdicia ancho de banda a causa de las colisiones. Los conmutadores almacenan las tramas en buffer y nunca transmiten más de una trama a un segmento simultáneamente. Al igual que con los routers, la tasa máxima de transferencia agregada de un conmutador es la suma de todas las tasas de las interfaces del conmutador. Por tanto, los conmutadores proporcionan una mejora significativa en cuanto al rendimiento respecto al de las redes LAN con enlaces de difusión.

- *Enlaces heterogéneos*. Dado que un conmutador aísla un enlace de otro, los distintos enlaces de una LAN pueden operar a velocidades diferentes y pueden utilizar diferentes medios físicos. Por ejemplo, en la Figura 5.24, A puede conectarse mediante un enlace de cobre 10BASE-T a 10 Mbps, B puede conectarse mediante un enlace de fibra 100BASE-FX a 100 Mbps y C puede conectarse mediante un enlace de cobre 1000 BASE-T a 1 Gbps. Por tanto, un conmutador es ideal para combinar equipos heredados con equipos nuevos.
- *Administración*. Además de proporcionar una seguridad mejorada (véase el recuadro dedicado a la seguridad), un conmutador también facilita las tareas de gestión de la red. Por ejemplo, si un adaptador de red funciona mal y envía continuamente tramas Ethernet, un conmutador puede detectar el problema y desconectar internamente el adaptador que está funcionando incorrectamente. Con esta característica, el administrador de la red no tiene que levantarse de la cama y conducir hasta la oficina para corregir el problema. Del mismo modo, un corte en un cable sólo desconecta al nodo que está usando el cable cortado para conectarse al conmutador. En la época del cable coaxial, los administradores de red pasaban muchas horas "recorriendo las líneas" (o dicho de forma más precisa, "arrastrándose por el suelo") hasta localizar el cable roto que había hecho que se cayera toda la red. Como veremos en el Capítulo 9 (Administración de la red), los conmutadores también recopilan estadísticas acerca del uso del ancho de banda, de las tasas de colisión y de los tipos de tráfico, y ponen esta información a disposición del administrador de la red. Esta información puede emplearse para depurar y corregir problemas y para planificar cómo deberá la red LAN evolucionar en el futuro. Los investigadores están explorando la adición de todavía más funcionalidades para la administración de las redes LAN Ethernet en implantaciones de prototipos [Casado 2007].

5.6.4 Conmutadores frente a routers

Como hemos visto en el Capítulo 4, los routers son dispositivos de conmutación de almacenamiento y reenvío que reenvían los paquetes utilizando direcciones de la capa de red. Aunque un conmutador también es un dispositivo de conmutación de paquetes de almacenamiento y reenvío, es fundamentalmente diferente de un router porque reenvía los paquetes utilizando direcciones MAC. Mientras que un router es un dispositivo de conmutación de paquetes de la capa 3, un conmutador es un dispositivo de conmutación de paquetes de la capa 2.

Aunque los conmutadores y los routers son fundamentalmente diferentes, los administradores de red a menudo tienen que elegir entre ellos a la hora de instalar un dispositivo de interconexión. Por ejemplo, en la red de la Figura 5.26 el administrador de red podría haber utilizado fácilmente un router en lugar de un conmutador para conectar las redes LAN departamentales, los servidores y el router de pasarela de Internet. De hecho, un router permitiría las comunicaciones entre departamentos sin crear colisiones. Puesto que tanto los conmutadores como los routers son candidatos como dispositivos de interconexión, ¿cuáles son los pros y los contras de cada uno de ellos?

En primer lugar vamos a ocuparnos de los pros y los contras de los conmutadores. Como hemos mencionado anteriormente, los conmutadores son dispositivos plug-and-play, una propiedad muy apreciada por todos los administradores de red del mundo sobrecargados de trabajo. Los conmutadores también ofrecen tasas de filtrado y reenvío relativamente altas (como se muestra en la Figura 5.29, los conmutadores tienen que procesar las tramas

SEGURIDAD

HUSMEANDO EN UNA LAN CONMUTADA: ENVENENAMIENTO DE UN CONMUTADOR

Cuando un nodo está conectado a un conmutador, normalmente sólo recibe las tramas que le están siendo enviadas de forma explícita. Por ejemplo, considere la red LAN conmutada de la Figura 5.24. Cuando el nodo A envía una trama al nodo B y existe una entrada para el nodo B en la tabla del conmutador, éste reenviará la trama *únicamente* al nodo B. Si el nodo C está ejecutando un programa husmeador (*sniffer*), no podrá husmear esta trama de A a B. Por tanto, en una LAN conmutada (en contraste con un entorno de enlaces de difusión como las redes LAN 802.11 o las redes LAN Ethernet basadas en concentrador) es más difícil para un atacante husmear las tramas. Sin embargo, dado que los conmutadores difunden las tramas que tienen direcciones de destino que no están almacenadas en sus tablas de conmutación, el programa sniffer de C puede todavía husmear algunas tramas que no están explícitamente dirigidas a C. Además, un programa sniffer podrá husmear todas las tramas Ethernet de difusión que tengan la dirección de destino de difusión FF–FF–FF–FF–FF–FF. Un ataque bien conocido contra un conmutador, que recibe el nombre de **envenenamiento de conmutador**, consiste en enviar toneladas de paquetes al conmutador con muchas direcciones MAC de origen falsas diferentes, que rellenarán la tabla del conmutador con entradas falsas y no dejarán espacio para las direcciones MAC de los nodos legítimos. Esto hace que el conmutador difunda la mayor parte de las tramas, las cuales pueden ser seleccionadas por el husmeador [Skoudis 2006]. Dado que este ataque es bastante complejo incluso para un atacante avanzado, los conmutadores son significativamente menos vulnerables a los *sniffers* que las redes LAN inalámbricas y basadas en concentradores.

sólo hasta la capa 2, mientras que los routers tienen que procesar los datagramas hasta la capa 3). Por otro lado, para impedir los ciclos de las tramas de difusión, la topología activa de una red conmutada está restringida a un árbol de recubrimiento. Además, una red conmutada grande requerirá tablas ARP grandes en los nodos y generará una cantidad de procesamiento y tráfico ARP sustancial. Los conmutadores no ofrecen ninguna protección frente a las tormentas de difusión (si un host está descontrolado y transmite un flujo de tramas Ether-

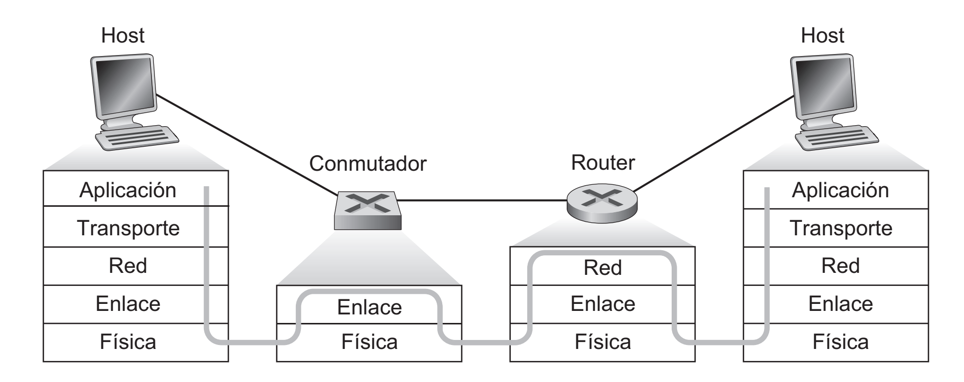


Figura 5.29 • Procesamiento de paquetes en conmutadores, routers y hosts.

net de difusión sin fin, los conmutadores reenviarán todas esas tramas, haciendo que toda la red colapse).

Consideremos ahora los pros y los contras de los routers. Puesto que frecuentemente el direccionamiento de red es jerárquico (y no plano como el direccionamiento MAC), normalmente los paquetes no seguirán ciclos a través de los routers incluso cuando la red tenga rutas redundantes. (Sin embargo, los paquetes pueden seguir ciclos cuando las tablas del router están mal configuradas; pero como hemos estudiado en el Capítulo 4, IP utiliza un campo especial de la cabecera del datagrama para limitar estos ciclos.) Por tanto, los paquetes no están restringidos a un árbol de recubrimiento y pueden utilizar la mejor ruta entre el origen y el destino. Dado que los routers no tienen la restricción del árbol de recubrimiento, han permitido que Internet haya sido creada con una topología rica que incluye, por ejemplo, múltiples enlaces activos entre Europa y América del Norte. Otra funcionalidad de los routers es que proporcionan protección mediante cortafuegos frente a las tormentas de difusión de la capa 2. Quizá el inconveniente más significativo de los routers es que no son dispositivos plug-and-play (ellos y los hosts conectados a ellos necesitan que sus direcciones IP sean configuradas). Además, los routers suelen tener un tiempo de procesamiento por paquete mayor que los conmutadores, ya que tienen que procesar campos hasta la capa 3. Por último, en inglés existen dos formas diferentes de pronunciar la palabra *router* ("*rootor*" o "*rowter*"), y la gente pierde mucho tiempo discutiendo acerca de la pronunciación apropiada [Perlman 1999].

Dado que tanto los conmutadores como los routers tienen sus ventajas y sus inconvenientes (como se resume en la Tabla 5.1), entonces ¿cuándo debe utilizar una red institucional (por ejemplo, un red de un campus universitario o una red corporativa) conmutadores y cuándo routers? Normalmente, las redes pequeñas que constan de unos pocos cientos de hosts tienen menos segmentos de LAN. Los conmutadores son suficientes para estas redes pequeñas, ya que localizan el tráfico y aumentan la tasa de transferencia agregada sin la necesidad de configurar direcciones IP. Pero las redes de mayor tamaño que constan de miles de hosts suelen incluir routers dentro de la red (además de conmutadores). Los routers proporcionan un aislamiento más robusto del tráfico, controlan las tormentas de difusión y utilizan rutas más "inteligentes" entre los hosts de la red.

Para ver una exposición sobre los pros y los contras de las redes conmutadas y enrutadas, así como un estudio de cómo la tecnología LAN conmutada puede ampliarse para acomodar dos órdenes de magnitud más de hosts que las redes Ethernet actuales, consulte [Kim 2008].

5.6.5 Redes de área local virtuales (VLAN)

En nuestra explicación anterior de la Figura 5.26 hemos mencionado que las redes LAN institucionales modernas suelen estar configuradas de forma jerárquica, teniendo cada grupo de trabajo (departamento) su propia red LAN conmutada conectada a las redes LAN conmutadas de los otros grupos a través de una jerarquía de conmutadores. Aunque una configuración de este tipo funciona bien en un mundo ideal, el mundo real está bastante alejado del ideal. En la configuración de la Figura 5.26 podemos identificar tres desventajas:

- *Falta de aislamiento del tráfico.* Aunque la jerarquía localiza el tráfico del grupo dentro de un mismo conmutador, el tráfico de difusión (por ejemplo, las tramas que transportan los mensajes ARP y DHCP o las tramas cuyo destino todavía no ha sido aprendido por un conmutador auto-aprendiz) tienen que atravesar toda la red institucional. Limitar el

	Concentradores	Routers	Conmutadores
Aislamiento del tráfico	No	Sí	Sí
Plug and play	Sí	No	Sí
Enrutamiento óptimo	No	Sí	No

Tabla 5.1 • Comparación de las funcionalidades típicas de dispositivos de interconexión populares.

ámbito del tráfico de difusión mejoraría el rendimiento de la LAN. Quizá más importante, sería también deseable limitar el tráfico de difusión de la LAN por razones de seguridad y confidencialidad. Por ejemplo, si un grupo contiene al equipo de dirección de la empresa y otro grupo tiene empleados descontentos que ejecutan paquetes sniffer Wireshark, probablemente el administrador de la red preferirá que el tráfico del equipo de dirección nunca llegue a los hosts de los empleados. Este tipo de aislamiento podría proporcionarse sustituyendo el conmutador central de la Figura 5.26 por un router. Enseguida veremos que este aislamiento también se puede conseguir a través de una solución conmutada (capa 2).

- *Uso ineficiente de los conmutadores.* Si en lugar de tres grupos la institución tiene 10 grupos, entonces se necesitarían 10 conmutadores de primer nivel. Si cada uno de los grupos es pequeño, por ejemplo, están formados por menos de 10 personas, entonces un único conmutador de 96 puertos sería lo suficientemente grande como para acomodar a todo el mundo, pero este único conmutador no proporcionaría la funcionalidad de aislamiento del tráfico.
- *Gestión de los usuarios.* Si un empleado se mueve entre grupos, el cableado físico debe modificarse para conectar al empleado a un conmutador diferente de la Figura 5.26. Los empleados que pertenecen a dos grupos constituyen incluso un problema mayor.

Afortunadamente, cada una de estas desventajas puede ser abordada por un conmutador compatible con **redes de área local virtuales** (**VLAN,** ***Virtual Local Area Network***).

Como su nombre sugiere, un conmutador compatible con redes VLAN permite definir múltiples redes de área local *virtuales* sobre una única infraestructura de red de área local *física*. Los hosts de una VLAN se comunican entre sí como si sólo ellos (y ningún otro host) estuvieran conectados al conmutador. En una VLAN basada en puertos, el administrador de la red divide los puertos (interfaces) del conmutador en grupos. Cada grupo constituye una VLAN, con los puertos de cada VLAN formando un dominio de difusión (es decir, el tráfico de difusión de un puerto sólo puede llegar a los demás puertos del grupo). La Figura 5.30 muestra un único conmutador con 16 puertos. Los puertos 2 a 8 pertenecen a la VLAN IE, y los puertos 9 a 15 pertenecen a la VLAN CC (los puertos 1 y 16 no están asignados). Esta VLAN resuelve todas las dificultades mencionadas anteriormente (las tramas de las VLAN IE y CC están aisladas entre sí, los dos conmutadores de la Figura 5.26 se han sustituido por un único conmutador y si el usuario del puerto 8 del conmutador se une al departamento CC, el operador de red simplemente tendrá que reconfigurar el software de la VLAN de modo que el puerto 8 ahora esté asociado con la VLAN CC. Es fácil imaginar cómo se configura y funciona el conmutador para redes VLAN: el administrador de la red declara que un puerto pertenece a una determinada VLAN (los puertos no declarados pertenecen a una VLAN

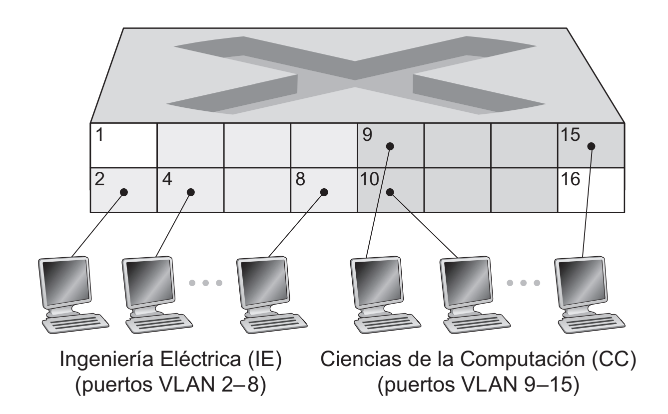


Figura 5.30 • Un mismo conmutador con dos redes VLAN configuradas.

predeterminada) utilizando un software de gestión de conmutadores; en el conmutador se mantiene una tabla de correspondencias entre puertos y redes VLAN y el hardware del conmutador sólo entrega tramas entre puertos que pertenecen a la misma VLAN.

Pero a causa del completo aislamiento de las dos redes VLAN, hemos introducido una nueva dificultad: ¿cómo puede enviarse el tráfico del departamento IE al departamento CC? Una forma de resolver esto sería conectando un puerto del conmutador VLAN (por ejemplo, el puerto 1 en la Figura 5.30) a un router externo y configurando dicho puerto para que pertenezca tanto a la VLAN IE como a la VLAN CC. En este caso, incluso aunque los departamentos IE y CC compartan el mismo conmutador físico, la configuración lógica sería como si dichos departamentos tuvieran conmutadores separados conectados a través de un router. Un datagrama IP enviado desde el departamento IE al departamento CC primero atravesaría la VLAN IE para llegar al router y luego sería reenviado por el router por la VLAN CC hasta el host de CC. Afortunadamente, los fabricantes de conmutadores hacen que dicha tarea de configuración resulte sencilla para los administradores de red, incorporando en un único dispositivo un conmutador VLAN *y* un router, con lo que no es necesario utilizar un router externo separado. En uno de los problemas de repaso del final del capítulo se examina este escenario más en detalle.

Volvamos de nuevo a la Figura 5.26. Suponga ahora que en lugar de tener un departamento de Ingeniería Informática separado, algunos de los académicos de IE y CC están alojados en edificios diferentes, donde (¡por supuesto!) necesitan tener acceso a la red y (¡por supuesto también!) desean formar parte de la VLAN de su departamento. La Figura 5.31 muestra un segundo conmutador de 8 puertos, en el que los puertos se han definido como pertenecientes a la VLAN IE o a la VLAN CC, según sea necesario. Pero, ¿cómo deben interconectarse estos dos conmutadores? Una solución fácil sería definir un puerto que perteneciera a la VLAN CC en cada conmutador (y lo mismo para la VLAN IE) y conectar estos puertos entre sí, como se muestra en la Figura 5.31(a). Sin embargo, esta solución no es escalable, ya que *N* redes VLAN requerirían *N* puertos en cada conmutador simplemente para interconectar los dos conmutadores.

Un método más escalable consiste en interconectar los conmutadores VLAN utilizando la técnica conocida como **troncalización VLAN (VLAN *Trunking*)**. Con esta técnica, mostrada en la Figura 5.31(b), un puerto especial de cada conmutador (el puerto 16 en el conmutador de la izquierda y el puerto 1 en el de la derecha) se configura como un puerto

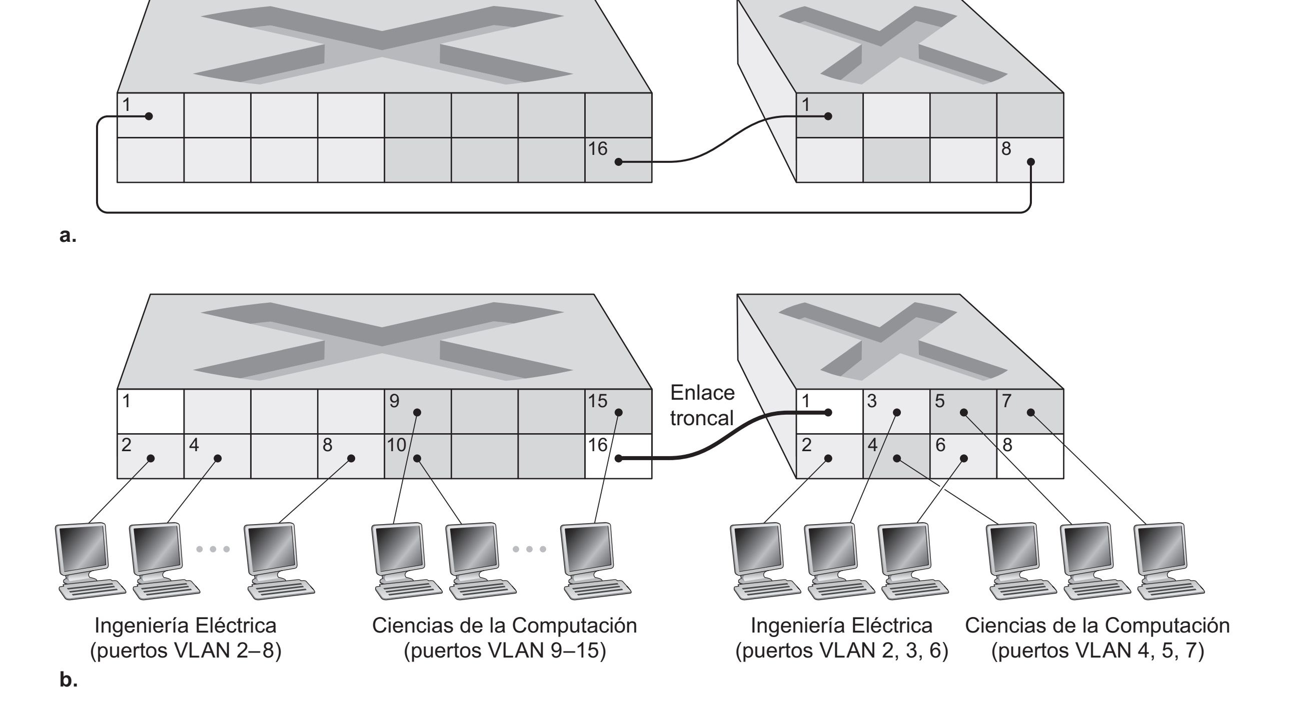


Figura 5.31 • Conexión de dos conmutadores VLAN con dos redes VLAN: (a) dos cables (b) enlace troncal.

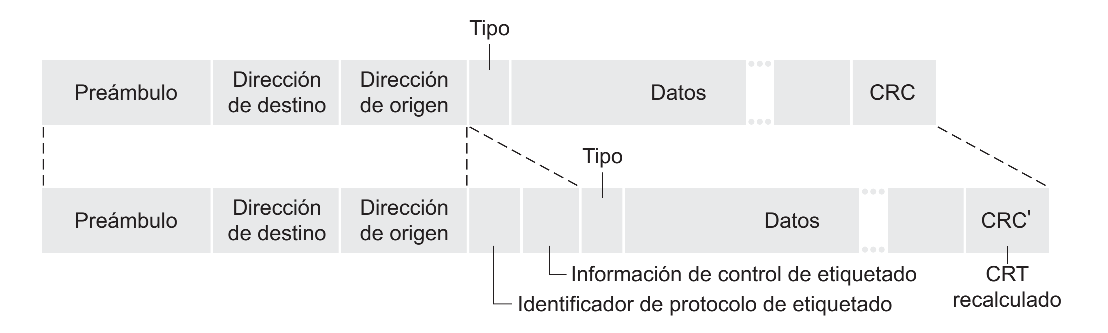


Figura 5.32 • Trama Ethernet original (arriba), trama Ethernet con etiquetado VLAN 802.1Q (abajo).

troncal para interconectar los dos conmutadores VLAN. El puerto troncal pertenece a todas las VLAN y las tramas enviadas a cualquier VLAN son reenviadas a través del enlace troncal hacia el otro conmutador. Pero esta solución nos conduce a otra pregunta: ¿cómo sabe un conmutador que una trama que ha llegado a un puerto troncal pertenece a una VLAN concreta? El IEEE ha definido un formato de trama Ethernet ampliado, 802.1Q, para las tramas que atraviesan un enlace troncal VLAN. Como se muestra en la Figura 5.32, la trama 802.1Q está formada por la trama Ethernet estándar más una **etiqueta VLAN** de cuatro

bytes añadida a la cabecera que transporta la identidad de la VLAN a la que pertenece la trama. El conmutador del lado emisor de un enlace troncal VLAN añade la etiqueta VLAN a la trama, la cual es analizada y eliminada por el conmutador del lado receptor del enlace troncal. La etiqueta VLAN en sí consta de un campo Identificador de protocolo de etiquetado (TPID, *Tag Protocol Identifier*) de 2 bytes (que tiene un valor hexadecimal fijo de 81-00), un campo Información de control de etiquetado (*Tag Control Information*) de 2 bytes, que contiene un campo identificador de VLAN de 12 bits y un campo de prioridad de 3 bits, cuya finalidad es similar a la del campo TOS de los datagramas IP.

En esta exposición hemos hablado muy brevemente de las redes VLAN y nos hemos centrado en las VLAN basadas en puertos. Debemos decir también que las redes VLAN pueden definirse de otras formas. En las VLAN basadas en direcciones MAC, el administrador de redes especifica el conjunto de direcciones MAC que pertenece a cada VLAN. Cuando un dispositivo se conecta a un puerto, el puerto se conecta a la VLAN apropiada basándose en la dirección MAC del dispositivo. Las redes VLAN también pueden definirse basándose en protocolos de la capa de red (como por ejemplo, IPv4, IPv6 o Appletalk) y en otros criterios. Consulte el estándar 802.1Q [IEEE 802.1q 2005] para conocer más detalles.

5.7 PPP: Protocolo punto a punto

Hasta el momento gran parte de nuestra exposición sobre los protocolos de la capa de enlace ha estado centrada en los protocolos para los canales de difusión. En esta sección vamos a abordar un protocolo de la capa de enlace para los enlaces punto a punto: PPP, el protocolo punto a punto. Puesto que PPP es típicamente el protocolo elegido para un enlace de acceso telefónico de un host residencial, es indudablemente uno de los protocolos de la capa de enlace más ampliamente implantado actualmente. El otro protocolo de la capa de enlace importante en uso hoy día es el protocolo HDLC (*High-level Data Link Control*); consulte [Spragins 1991] si desea obtener información acerca de HDLC. Aquí, nuestra exposición sobre el protocolo PPP más simple nos va a permitir examinar muchas de las más importantes funcionalidades de un protocolo de la capa de enlace punto a punto.

Como su nombre implica, el protocolo punto a punto PPP [RFC 1661; RFC 2153] es un protocolo de la capa de enlace que opera sobre un **enlace punto a punto**: un enlace que conecta directamente dos nodos situados cada uno de ellos en un extremo del enlace. El enlace punto a punto sobre el que PPP opera puede ser una línea telefónica serie (por ejemplo, una conexión por módem de 56K), un enlace SONET/SDH, una conexión X.25 o un circuito RDSI. Como ya hemos mencionado, PPP suele ser el protocolo elegido por los usuarios domésticos para conectarse con sus ISP a través de una conexión de acceso telefónico.

Antes de profundizar en los detalles de PPP, es interesante examinar los requisitos originales que el IETF estableció para el diseño de PPP [RFC 1547]:

- *Entramado de paquetes.* El emisor de la capa de enlace del protocolo PPP tiene que poder tomar un paquete del nivel de red y encapsularlo dentro de la trama de la capa de enlace PPP de tal modo que el receptor sea capaz de identificar el inicio y el final tanto de la trama de la capa de enlace como del paquete de la capa de red contenido en ella.
- *Transparencia.* El protocolo PPP no debe aplicar ninguna restricción a los datos que aparecen en el paquete de la capa de red (ni a las cabeceras ni a los datos). Por ejemplo, PPP

no puede prohibir el uso de ciertos patrones de bits en el paquete de la capa de red. Volveremos sobre este tema enseguida cuando hablemos del rellenado de bytes.

- *Múltiples protocolos de la capa de red.* El protocolo PPP tiene que poder dar soporte a múltiples protocolos de la capa de red (por ejemplo, IP y DECnet) que se ejecuten sobre el mismo enlace físico de forma simultánea. Del mismo modo que se necesita el protocolo IP para multiplexar diferentes protocolos de la capa de transporte (por ejemplo, TCP y UDP) sobre una única conexión terminal a terminal, también PPP tiene que ser capaz de multiplexar diferentes protocolos de la capa de red sobre una única conexión punto a punto. Este requisito quiere decir que, como mínimo, probablemente PPP requerirá un campo de tipo de protocolo o algún mecanismo similar para que el PPP del lado receptor pueda demultiplexar una trama recibida en el protocolo de la capa de red apropiado.
- *Múltiples tipos de enlaces.* Además de poder transportar múltiples protocolos de nivel superior, PPP también tiene que poder operar sobre una amplia variedad de tipos de enlaces, incluyendo enlaces serie (que transmiten un bit cada vez en una dirección dada) o paralelo (que transmiten bits en paralelo), enlaces síncronos (que transmiten una señal de reloj junto con los bits de datos) o asíncronos, enlaces de baja velocidad o de alta velocidad, o enlaces eléctricos u ópticos.
- *Detección de errores.* Un receptor PPP tiene que ser capaz de detectar errores de bit en las tramas recibidas.
- *Pervivencia de la conexión.* PPP tiene que ser capaz de detectar un fallo en el nivel de enlace (por ejemplo, la incapacidad para transferir datos desde el lado emisor del enlace al lado receptor del mismo) y de señalar esa condición de error a la capa de red.
- *Negociación de direcciones de la capa de red.* PPP tiene que proporcionar un mecanismo para que las capas de red (por ejemplo, IP) que se están comunicando puedan aprender o configurar las direcciones de la capa de red de cada una de ellas.
- *Simplicidad.* Se requirió que PPP cumpliera una serie de requisitos adicionales además de los que acabamos de enumerar. El primero y más importante de todos ellos es la simplicidad. El documento RFC 1547 establece que "El lema de un protocolo punto a punto debe ser la simplicidad." ¡Una orden muy estricta, teniendo en cuenta todos los demás requisitos definidos para el diseño de PPP! Aproximadamente 100 documentos RFC definen ahora los distintos aspectos de este protocolo "simple".

Aunque puede parecer que son demasiados los requisitos impuestos al diseño de PPP, la situación podría haber sido realmente más difícil. Las especificaciones de diseño de PPP también indican de forma explícita las funcionalidades de protocolo que PPP *no* requería implementar:

- *Corrección de errores.* Se requiere que PPP detecte los errores de bit pero *no* se requiere que los corrija.
- *Control de flujo.* Se espera que un receptor PPP sea capaz de recibir tramas a la velocidad máxima de la capa física subyacente. Si una capa superior no puede recibir paquetes a esa velocidad máxima, entonces es responsabilidad de la capa superior el eliminar paquetes o regular al emisor en la capa superior. Es decir, en lugar de que el propio emisor PPP regule su velocidad de transmisión, es responsabilidad de un protocolo de nivel superior el regular la velocidad a la que los paquetes son entregados a PPP para su transmisión.

- *Secuenciamiento. No* se requiere que PPP entregue las tramas al receptor en el mismo orden en que fueron enviadas por el emisor. Es interesante observar que aunque esta flexibilidad es compatible con el modelo de servicio de IP (que permite que los paquetes IP sean entregados terminal a terminal en cualquier orden), otros protocolos de la capa de red que operan sobre PPP requieren una entrega de paquetes terminal a terminal en secuencia.
- *Enlaces multipunto.* PPP sólo necesita operar sobre enlaces que tienen un único emisor y un único receptor. Otros protocolos de la capa de enlace, como HDLC, pueden acomodar varios receptores en un enlace (por ejemplo, un escenario tipo Ethernet).

Una vez vistos los objetivos de diseño (y los no objetivos) de PPP, pasemos a ver cómo cumple el diseño de PPP dichos objetivos.

5.7.1 Trama de datos PPP

La Figura 5.33 muestra una trama de datos PPP que utiliza un entramado tipo HDLC [RFC 1662]. La trama PPP consta de los campos siguientes:

- *Campo Indicador.* Todas las tramas PPP comienzan y terminan con un campo indicador de 1 byte, cuyo valor es 01111110.
- *Campo de dirección.* El único valor posible en este campo es 11111111.
- *Campo de control.* El único valor posible en este campo es 00000011. Puesto que los campos de dirección y de control sólo pueden tomar un valor fijo, es posible que se esté preguntando por qué se han definido dichos campos. La especificación de PPP [RFC 1662] establece que otros valores "pueden ser definidos más adelante", aunque no se ha definido ninguno hasta la fecha. Dado que estos valores toman valores fijos, PPP permite al emisor no enviar los bytes de dirección y de control, ahorrando así 2 bytes de sobrecarga en la trama PPP.
- *Protocolo.* El campo protocolo indica al receptor PPP el protocolo de la capa superior al que pertenecen los datos encapsulados recibidos (es decir, el contenido del campo de información de la trama PPP). Al recibir una trama PPP, el receptor PPP comprobará si la trama es correcta y luego pasará los datos encapsulados al protocolo apropiado. Los documentos RFC 1700 y RFC 3232 definen los códigos de protocolo de 16 bits utilizados por PPP. A nosotros nos interesa el protocolo IP (es decir, los datos encapsulados en la trama PPP forman un datagrama IP), el cual tiene un valor hexadecimal de 21; aunque también pueden emplearse otros protocolos de la capa de red como AppleTalk, que tiene el valor 29, y el protocolo DECnet, que tiene el valor 27.

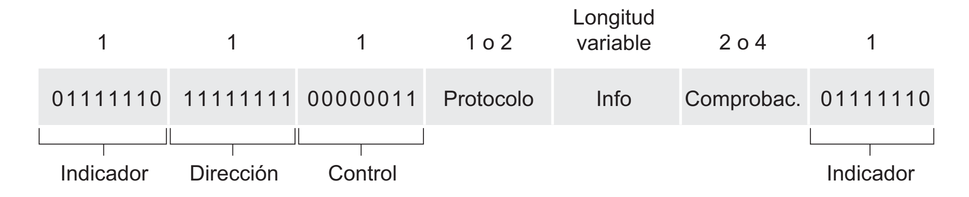


Figura 5.33 • Formato de la trama de datos PPP.

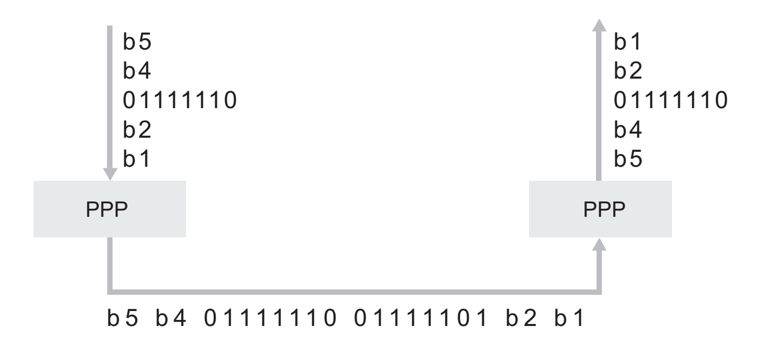


Figura 5.34 • Rellenado de bytes.

- *Información.* Este campo contiene el paquete (datos) encapsulado que está siendo enviado por un protocolo de la capa superior (por ejemplo, IP) sobre el enlace PPP. La longitud máxima predeterminada de este campo es 1.500 bytes, aunque como veremos más adelante este valor puede cambiarse cuando se configura el enlace por primera vez.
- *Suma de comprobación.* El campo de suma de comprobación se utiliza para detectar errores de bit en una trama transmitida. Se emplea un código de redundancia cíclica estándar HDLC de 2 o 4 bytes.

Rellenado de bytes

Antes de terminar nuestra exposición acerca de las tramas PPP, consideremos un problema que surge cuando cualquier protocolo utiliza un patrón de bits específico en un campo indicador para marcar el principio o el final de la trama. ¿Qué ocurre si el patrón indicador aparece en cualquier posición del paquete? Por ejemplo, ¿qué ocurre si el valor del campo indicador 01111110 aparece en el campo de información? ¿Detectará el receptor incorrectamente el final de la trama PPP?

Una forma de resolver este problema sería que PPP prohibiera al protocolo de la capa superior que enviara datos que contuvieran el patrón de bits del campo indicador. El requisito de transparencia de PPP mencionado anteriormente elimina esta posibilidad. Una solución alternativa, que es la que emplea PPP y otros muchos protocolos, consiste en utilizar una técnica conocida como **rellenado de bytes**.

PPP define un byte especial de escape de control, 01111101. Si la secuencia del indicador, 01111110, aparece en cualquier posición de la trama, excepto en el campo indicador, PPP precede dicha aparición del patrón indicador con el byte de escape de control. Es decir, "rellena" (añade) con un byte de escape de control el flujo de datos transmitidos que precede a 01111110, con el fin de indicar que el siguiente 011111110 *no* es un valor de indicador sino que son datos reales. Por supuesto, un receptor que ve una secuencia 01111110 precedida por 01111101 eliminará el byte de escape de control añadido para reconstruir los datos originales. De forma similar, si el propio patrón de bits del byte de escape de control aparece como datos reales, también deberá anteponérsele el byte de escape de control de relleno. Por tanto, cuando el receptor ve un único byte de escape de control en el flujo de datos, sabe que el byte fue añadido al flujo de datos. Dos bytes de escape de control seguidos quieren decir que uno de ellos pertenece a los datos originales que están siendo enviados. La Figura 5.34 ilustra la técnica de relleno de bytes de PPP. (Realmente, PPP también aplica una operación XOR con

el hexadecimal 20 al byte de datos al que le precede el byte de escape, un detalle que hemos omitido para simplificar la exposición.)

Debemos comentar también que PPP tiene un Protocolo de control de enlace (LCP, *Link Control Protocol*), cuyo trabajo consiste en llevar a cabo la inicialización, el mantenimiento y el cierre de un enlace PPP. LCP se estudia con cierto detalle en el material en línea asociado con el libro.

5.8 Virtualización de enlaces: la red como una capa de enlace

Puesto que este capítulo está relacionado con los protocolos de la capa de enlace, y dado que estamos aproximándonos al final del capítulo, vamos a reflexionar acerca del modo en que ha evolucionado nuestra compresión acerca del término *enlace.* Al comenzar el capítulo, contemplábamos los enlaces como cables físicos que conectaban dos hosts que se comunicaban entre sí, como se ilustraba en la Figura 5.2. Al estudiar los protocolos de acceso múltiple (Figura 5.9), hemos visto que podían conectarse varios hosts mediante un cable compartido y que el "cable" que conectaba esos hosts podía ser un espectro de radio u otro medio de comunicación. Esto nos llevó a considerar el enlace en forma algo más abstracta como un canal más que como un cable. En nuestro estudio de las redes LAN Ethernet (Figuras 5.26 y 5.31), hemos visto que los medios de interconexión podían ser, de hecho, una infraestructura conmutada bastante compleja. Sin embargo, a todo lo largo de esta evolución, los propios hosts mantenían la visión de que el medio de interconexión era simplemente un canal de la capa enlace que conectaba dos o más hosts. Vimos, por ejemplo, que un host Ethernet podía ser afortunadamente inconsciente de si estaba conectado a los hosts de otra LAN mediante un único segmento LAN de corto alcance (Figura 5.9) o por una LAN conmutada geográficamente dispersa (Figura 5.26) o mediante una VLAN (Figura 5.31).

En la Sección 5.7 hemos visto que el protocolo PPP se emplea a menudo sobre una conexión vía módem entre dos hosts. En este caso, el enlace que conecta los dos hosts es realmente la red telefónica: una red global de telecomunicaciones, lógicamente separada, con sus propios conmutadores, enlaces y pilas de protocolos para la transferencia de datos y la señalización. Sin embargo, desde el punto de vista de la capa de enlace de Internet, la conexión de acceso telefónico a través de la red de telefonía es contemplada como un simple "cable". En este sentido, Internet virtualiza la red telefónica viéndola como una tecnología de la capa de enlace que proporciona conectividad de dicha capa entre dos hosts de Internet. Recordemos, de nuestro análisis de las redes solapadas en el Capítulo 2, que de forma similar las redes solapadas ven Internet como un medio de proporcionar conectividad entre los nodos solapados, buscando un solapamiento de Internet de la misma manera que Internet solapa la red telefónica.

En esta sección vamos a abordar las redes de Conmutación de etiquetas multiprotocolo (MPLS, *Multiprotocol Label Switching*). A diferencia de la red telefónica de conmutación de circuitos, MPLS es una red de circuitos virtuales de conmutación de paquetes de pleno derecho. Tiene sus propios formatos de paquete y comportamientos de reenvío. Por tanto, desde el punto de vista pedagógico, el análisis de MPLS encaja bien en un estudio de la capa de red o de la capa de enlace. Sin embargo, desde el punto de vista de Internet, podemos considerar MPLS, al igual que la red telefónica y las redes Ethernet conmutadas, como

una tecnología de la capa de enlace que sirve para interconectar dispositivos IP. Por tanto, hemos incluido MPLS en nuestro estudio de la capa de enlace. Las redes Frame-Relay y ATM también pueden emplearse para interconectar dispositivos IP, aunque representan una tecnología algo más antigua (aunque todavía con implantación), de modo que no cubriremos esas redes aquí. Puede ver más detalles en el libro de [Goralski 1999]. Nuestro tratamiento de MPLS será necesariamente breve, ya que podrían escribirse (y se han escrito) libros completos sobre estas redes. Le recomendamos que lea [Davie 2000] para conocer más detalles sobre MPLS. Aquí nos centraremos principalmente en el modo en que los servidores MPLS se interconectan con los dispositivos IP, aunque también profundizaremos algo más en las tecnologías subyacentes.

Conmutación de etiquetas multiprotocolo (MPLS)

La conmutación de etiquetas multiprotocolo (MPLS) ha evolucionado a partir de una serie de desarrollos industriales que tuvieron lugar a mediados y finales de la década de 1990 y que buscaban mejorar la velocidad de reenvío de los routers IP, adoptando un concepto clave del mundo de las redes de circuitos virtuales: una etiqueta de longitud fija. El objetivo no era abandonar la infraestructura de reenvío de datagramas IP basada en el destino, sustituyéndola por otra basada en etiquetas de longitud fija y circuitos virtuales, sino expandir la infraestructura existente etiquetando selectivamente los datagramas y permitiendo a los routers reenviar esos datagramas basándose en etiquetas de longitud fija (en lugar de en direcciones IP de destino), siempre que fuera posible. Es importante observar que estas técnicas funcionan mano a mano con IP, utilizando el direccionamiento y el enrutamiento IP. El IETF unificó estos esfuerzos mediante el protocolo MPLS [RFC 3031, RFC 3032], consiguiendo así mezclar de forma efectiva las técnicas de circuitos virtuales en una red de datagramas enrutados.

Comencemos nuestro estudio sobre MPLS considerando el formato de una trama de la capa de enlace gestionada por un router compatible con MPLS. La Figura 5.35 muestra que una trama de la capa de enlace transmitida a través de un enlace PPP o red LAN (como por ejemplo Ethernet) tiene una cabecera MPLS pequeña, que se añade entre la cabecera de la capa 2 (es decir, PPP o Ethernet) y la cabecera de la capa 3 (es decir, IP). El documento RFC 3032 define el formato de la cabecera MPLS para tales enlaces; en otros RFC se definen también las cabeceras para las redes ATM y Frame-Relay. Entre los campos de la cabecera MPLS se encuentran la etiqueta (que desempeña el papel del identificador de circuito virtual que hemos visto en la Sección 4.2.1); 3 bits reservados para su uso experimental; un único bit S, que se utiliza para indicar el final de una serie de cabeceras MPLS "apiladas" (un tema avanzado del que no hablaremos aquí) y un campo de tiempo de vida.

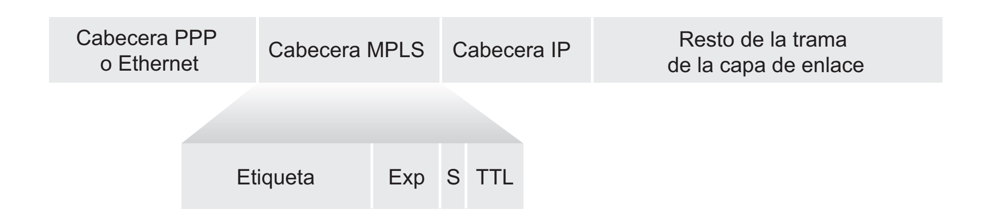
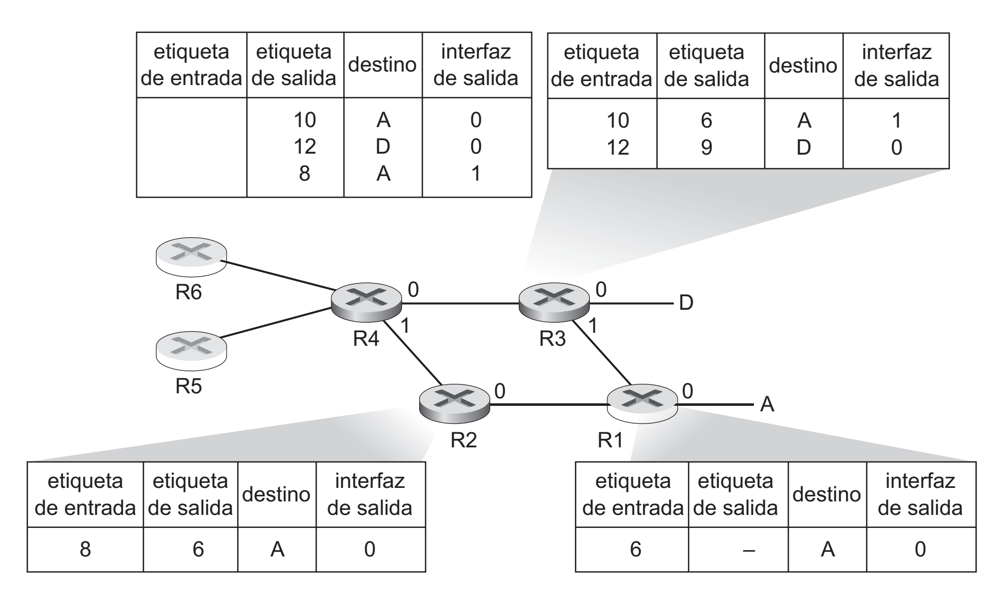


Figura 5.35 • Cabecera MPLS: se localiza entre las cabeceras de la capa de enlace y de la capa de red.

Es evidente de manera inmediata, a partir de la Figura 5.35, que una trama ampliada MPLS sólo puede intercambiarse entre routers compatibles con MPLS (dado que un router no compatible con MPLS se quedaría bastante confundido al encontrar una cabecera MPLS donde esperaba encontrar la cabecera IP). Los routers compatibles con MPLS se suelen denominar **routers de conmutación de etiquetas**, ya que reenvían las tramas MPLS buscando la etiqueta MPLS en su tabla de reenvío y luego pasando inmediatamente el datagrama a la interfaz de salida apropiada. Por tanto, el router compatible con MPLS *no* necesita extraer la dirección IP de destino y realizar una búsqueda del prefijo con la coincidencia más larga dentro de la tabla de reenvío. Pero, ¿cómo sabe un router si su vecino es compatible con MPLS y cómo sabe qué etiqueta asociar con la dirección IP de destino indicada? Para responder a estas preguntas, necesitamos examinar la interacción entre un grupo de routers compatibles con MPLS.

En el ejemplo de la Figura 5.36 los routers R1 a R4 son compatibles con MPLS. Los routers R5 y R6 son routers IP estándar. R1 ha anunciado a R2 y a R3 que puede enrutar hacia el destino A y que una trama recibida con una etiqueta MPLS igual a 6 será reenviada al destino A. El router R3 ha anunciado al router R4 que puede realizar el enrutamiento hacia los destinos A y D, y que las tramas entrantes con etiquetas MPLS de valor 10 y 12, respectivamente, serán conmutadas hacia esos destinos. El router R2 también ha anunciado al router R4 que puede alcanzar el destino A y que una trama recibida con la etiqueta MPLS de valor 8 será conmutada hacia A. Observe que ahora el router R4 se encuentra en la interesante situación de tener dos rutas MPLS para llegar a A: a través de la interfaz 0 con etiqueta MPLS saliente igual a 10, y a través de la interfaz 1 con etiqueta MPLS igual a 8. El cuadro general de la estructura de red mostrada en la Figura 5.36 es que los dispositivos IP R5, R6, A y D están interconectados a través de una infraestructura MPLS (los routers compatibles con MPLS R1, R2, R3 y R4), de forma muy similar a como pueden conectarse entre sí diversos dispositivos IP mediante una red ATM o una LAN conmutada. Y al igual que sucede con



etiqueta de entrada	etiqueta de salida	destino	interfaz de salida
	10	A	0
	12	D	0
	8	A	1

etiqueta de entrada	etiqueta de salida	destino	interfaz de salida
10	6	A	1
12	9	D	0

etiqueta de entrada	etiqueta de salida	destino	interfaz de salida
8	6	A	0

etiqueta de entrada	etiqueta de salida	destino	interfaz de salida
6	–	A	0

Figura 5.36 • Reenvío mejorado mediante MPLS.

una red ATM o una LAN conmutada, los routers R1 a R4 compatibles con MPLS se encargan de realizar esa conmutación *sin ni siquiera tocar la cabecera IP de los paquetes.*

En nuestra exposición anterior no hemos especificado el protocolo concreto que se utiliza para distribuir las etiquetas entre los routers compatibles con MPLS, ya que los detalles de este tipo de señalización caen fuera del alcance del libro. Sin embargo, tenemos que dejar constancia de que el grupo de trabajo de IETF dedicado a MPLS ha especificado en [RFC 3468] que el centro de sus esfuerzos dentro del campo de la señalización de MPLS será una extensión del protocolo RSVP (que estudiaremos en el Capítulo 7), conocida como RSVP-TE [RFC 3209]. Por tanto, el lector interesado puede consultar el documento RFC 3209.

Hasta ahora, el énfasis de nuestro estudio sobre MPLS se han centrado en el hecho de que MPLS realiza la conmutación basándose en etiquetas, sin necesidad de considerar la dirección IP de un paquete. Las verdaderas ventajas de MPLS y la razón del actual interés en este tipo de tecnología radica, sin embargo, no en los potenciales aumentos de las velocidades de conmutación, sino más bien en las nuevas capacidades de gestión del tráfico que MPLS posibilita. Como ya hemos indicado, R4 dispone de *dos* rutas MPLS hacia A. Si el reenvío se realizara más arriba, en la capa IP, basándose en la dirección IP, los protocolos de enrutamiento IP que hemos estudiado en el Capítulo 4 especificarían solamente una única ruta de coste mínimo hacia A. Por tanto, MPLS proporciona la capacidad de reenviar paquetes a través de rutas que no serían posibles utilizando los protocolos de enrutamiento IP estándar. Éste es un tipo simple de **ingeniería de tráfico** utilizando MPLS [RFC 3346; RFC 3272; RFC 2702; Xiao 2000], mediante el que un operador de red puede anular el enrutamiento IP normal y forzar a que parte del tráfico dirigido hacia un cierto destino tome una determinada ruta, mientras que el resto del tráfico dirigido a ese mismo destino sigue una ruta distinta (bien por razones de política, de rendimiento o de algún otro tipo).

También se puede utilizar MPLS para muchos otros propósitos. Puede emplearse para llevar a cabo una restauración rápida de las rutas de reenvío de MPLS; por ejemplo, para volver a enrutar el tráfico a través de una ruta de reserva precalculada como respuesta a un fallo de un enlace [Kar 2000; Huang 2002; RFC 3469]. MPLS también se puede emplear para implementar el marco de trabajo de servicio diferenciado que estudiaremos en el Capítulo 7. Por último, digamos que MPLS puede utilizarse (y de hecho se ha utilizado) para implementar las denominadas **redes privadas virtuales (VPN, *Virtual Private Network*)**. Al implementar una VPN para un cliente, un ISP utiliza su red compatible con MPLS para conectar entre sí las diversas redes del cliente. MPLS puede emplearse para aislar tanto los recursos como el direccionamiento empleados por la VPN del cliente con respecto a los de otros usuarios que también tengan que atravesar la red del ISP; consulte [DeClercq 2002] para ver más detalles.

Nuestra exposición acerca de MPLS ha sido necesariamente breve y animamos al lector a consultar las referencias mencionadas. Hay que destacar que, con tantos posibles usos de MPLS, este protocolo parece estar convirtiéndose rápidamente en la "navaja multiusos" de la ingeniería de tráfico en Internet.

5.9 Un día en la vida de una solicitud de página web

Ahora que ya hemos cubierto el tema de la capa de enlace en este capítulo, y las capas de red, de transporte y de aplicación en los anteriores, nuestro viaje de descenso por la pila de protocolos está completo. Al principio del libro (Sección 1.1), decíamos que "buena

parte de este libro está relacionada con los protocolos de las redes de computadoras", y en los primeros cinco capítulo hemos visto que es así. Antes de zambullirnos en los capítulos temáticos de la segunda parte del libro, conviene finalizar nuestro viaje descendente por la pila de protocolos adoptando una vista integrada y holística de los protocolos que hemos estudiado hasta el momento. Una forma, por tanto, de adoptar esta "vista panorámica" consiste en identificar los muchos (¡muchísimos!) protocolos implicados en satisfacer incluso la más simple de las solicitudes: la descarga de una página web. La Figura 5.37 ilustra el que será nuestro escenario de trabajo: un estudiante, Benito, conecta una computadora portátil al conmutador Ethernet de su facultad y descarga una página web (por ejemplo la página principal de www.google.com). Como sabemos ahora, son *muchas* las cosas que suceden "entre bambalinas" para satisfacer esta solicitud aparentemente simple. Una práctica de laboratorio con Wireshark incluida al final del capítulo le permitirá examinar con mayor detalle una serie de archivos de traza que contienen varios de los paquetes implicados en escenarios similares.

Inicio: DHCP, UDP, IP y Ethernet

Supongamos que Benito arranca su computadora portátil y luego la conecta a un cable Ethernet conectado al conmutador Ethernet de la facultad, que a su vez está conectado al router de la facultad, como se muestra en la Figura 5.37. El router de la facultad está conectado a un ISP, que en este caso se llama comcast.net. En este ejemplo, comcast.net proporciona el servicio DNS para la facultad; por tanto, el servidor DNS reside en la red de comcast, en lugar de en la red de la facultad. Supondremos que el servidor DHCP está ejecutándose dentro del router, como suele ser el caso.

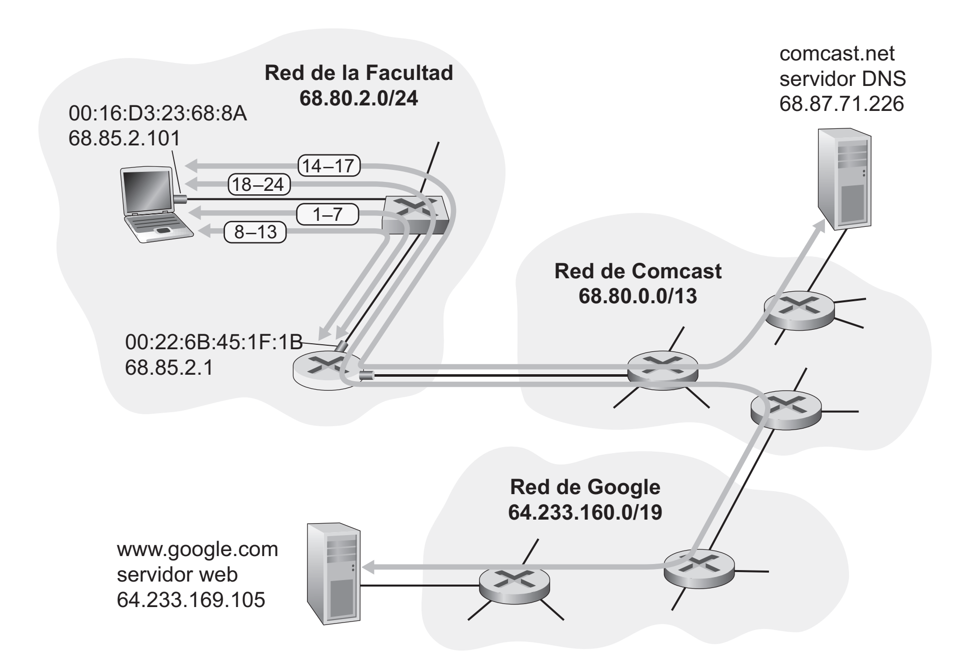


Figura 5.37

Cuando Benito conecta por primera vez su portátil a la red, no puede hacer nada (por ejemplo, descargar una página web) sin una dirección IP. Por tanto, la primera acción relacionada con la red que lleva a cabo la computadora portátil de Benito es ejecutar el protocolo DHCP para obtener una dirección IP, así como otras informaciones, desde el servidor DHCP local:

1. El sistema operativo del portátil de Benito crea un **mensaje de solicitud DHCP** (Sección 4.4.2) y lo incluye en un **segmento UDP** (Sección 3.3) con el puerto de destino 67 (servidor DHCP) y el puerto de origen 68 (cliente DHCP). A continuación, el segmento UDP es insertado dentro de un **datagrama IP** (Sección 4.4.1) con una dirección IP de destino de difusión (255.255.255.255) y una dirección IP de origen igual a 0.0.0.0, dado que la computadora portátil de Benito no tiene todavía una dirección IP.
2. El datagrama IP que contiene el mensaje de solicitud DHCP se inserta entonces en una **trama Ethernet** (Sección 5.5.1). La trama Ethernet tiene una dirección MAC de destino igual a FF:FF:FF:FF:FF:FF, de modo que la trama será difundida a todos los dispositivos conectados al conmutador (entre los que cabe esperar que se encuentre un servidor DHCP); la dirección MAC de origen de la trama es la de la computadora portátil de Benito, 00:16:D3:23:68:8A.
3. La trama Ethernet de difusión que contiene la solicitud DHCP es la primera trama enviada por la computadora de Benito al conmutador Ethernet. El conmutador difunde la trama entrante a todos los puertos de salida, incluyendo el puerto conectado al router.
4. El router recibe la trama Ethernet de difusión que contiene la solicitud DHCP a través de su propia interfaz con dirección MAC 00:22:6B:45:1F:1B, extrayendo el datagrama IP de la trama Ethernet. La dirección IP de destino de difusión contenida en el datagrama indica que este datagrama IP tiene que ser procesado por los protocolos de la capa superior existentes en este nodo. De modo que se **demultiplexa** (Sección 3.2) la carga útil del datagrama (un segmento UDP) y se entrega esa carga útil a UDP, tras lo cual se extrae del segmento UDP el mensaje de solicitud DHCP. Ahora el servidor DHCP dispone del mensaje de solicitud DHCP.
5. Supongamos que el servidor DHCP que se está ejecutando en el router puede asignar direcciones IP dentro del bloque **CIDR** (Sección 4.4.2) 68.85.2.0/24. En este ejemplo, todas las direcciones IP utilizadas dentro de la Facultad se encuentran dentro del bloque de direcciones de Comcast. Supongamos que el servidor DHCP asigna la dirección 68.85.2.101 al portátil de Benito. El servidor DHCP creará un **mensaje ACK DHCP** (Sección 4.4.2) que contendrá esta dirección IP, así como la dirección IP del servidor DNS (68.87.71.226), la dirección IP del router de pasarela predeterminado (68.85.2.1) y el bloque de subred (68.85.2.0/24) (o, lo que es lo mismo, la "máscara de red"). El mensaje DHCP se inserta dentro de un segmento UDP, que a su vez se incluye dentro de un datagrama IP, que se inserta en una trama Ethernet. La trama Ethernet tiene una dirección MAC de origen que será igual a la de la interfaz entre el router y la red doméstica (00:22:6B:45:1F:1B) y una dirección MAC de destino que será igual a la de la computadora portátil de Benito (00:16:D3:23:68:8A).
6. La trama Ethernet que contiene la respuesta ACK DHCP se envía (unidifusión) desde el router hacia el conmutador. Puesto que el conmutador tiene la característica de **autoaprendizaje** (Sección 5.6.2) y ha recibido anteriormente una trama Ethernet (la que

contenía la solicitud DHCP) desde el portátil de Benito, el conmutador sabe reenviar una trama dirigida a 00:16:D3:23:68:8A únicamente hacia el puerto de salida que conduce al portátil de Benito.

7. La computadora portátil de Benito recibe la trama Ethernet que contiene la respuesta ACK DHCP, extrae el datagrama IP de la trama Ethernet, extrae el segmento UDP del datagrama IP y extrae el mensaje ACK DHCP del segmento UDP. A continuación, el cliente DHCP de Benito anota su dirección IP y la dirección IP de su servidor DNS. También instala la dirección del router de pasarela predeterminado en su **tabla de reenvío IP** (Sección 4.1). El portátil de Benito enviará hacia el router de pasarela predeterminado todos los datagramas cuya dirección de destino caiga fuera de su subred 68.85.2.0/24. Llegados a este punto, la computadora portátil de Benito ha inicializado sus componentes de red y está lista para iniciar el procesamiento de la extracción de la página web. (Observe que sólo son necesarios los dos últimos pasos DHCP, de los cuatro presentados en el Capítulo 4.)

Seguimos con el inicio: DNS, ARP

Cuado Benito escribe la dirección URL correspondiente a www.google.com en su navegador web, comienza la larga cadena de sucesos que terminará por hacer que se muestre la página de inicio de Google en su navegador web. El navegador de Benito comienza creando un **socket TCP** (Sección 2.7) que se utilizará para enviar la **solicitud HTTP** (Sección 2.2) hacia www.google.com. Para crear el socket, la computadora portátil de Benito necesita conocer la dirección IP de www.google.com. En la Sección 2.5 hemos visto que se utiliza el **protocolo DNS** para proporcionar este servicio de traducción de nombres a direcciones IP.

8. El sistema operativo de la computadora de Benito crea por tanto un **mensaje de consulta DNS** (Sección 2.5.3), incluyendo la cadena "www.google.com" en la sección de consulta del mensaje DNS. Después, este mensaje DNS se inserta dentro de un segmento UDP con un puerto de destino igual a 53 (servidor DNS). Después el segmento UDP se inserta dentro de un datagrama IP con una dirección IP de destino igual a 68.87.71.226 (la dirección del servidor DNS devuelta en el mensaje ACK DHCP en el Paso 5) y una dirección IP de origen igual a 68.85.2.101.
9. La computadora portátil de Benito inserta entonces el datagrama que contiene el mensaje de consulta DNS dentro de una trama Ethernet. Esta trama será enviada (direccionada en la capa de enlace) al router de pasarela de la red de la facultad de Benito. Sin embargo, aún cuando el portátil de Benito conoce la dirección IP del router de pasarela de la facultad (68.85.2.1), gracias al mensaje ACK DHCP del Paso 5 anterior, no sabe la dirección MAC del router de pasarela. Para obtener la dirección MAC de este router de pasarela, el portátil de Benito necesita utilizar el **protocolo ARP** (Sección 5.4.2).
10. La computadora portátil de Benito crea un mensaje de **consulta ARP** con una dirección IP de destino igual a 68.85.2.1 (el router de pasarela predeterminado), incluye el mensaje ARP dentro de una trama Ethernet con una dirección de destino de difusión (FF:FF:FF:FF:FF:FF) y envía la trama Ethernet hacia el conmutador, que entrega la trama a todos los dispositivos conectados, incluyendo al router de pasarela.
11. El router de pasarela recibe la trama que contiene el mensaje de solicitud ARP a través de la interfaz con la red de la facultad y se encuentra con que la dirección IP de destino,

68.85.2.1, contenida en el mensaje ARP, coincide con la dirección IP de su propia interfaz. El router de pasarela prepara en consecuencia una **respuesta ARP**, indicando que su dirección MAC 00:22:6B:45:1F:1B se corresponde con la dirección IP 68.85.2.1. A continuación, inserta el mensaje de respuesta ARP en una trama Ethernet, con una dirección de destino igual a 00:16:D3:23:68:8A (la de la computadora portátil de Benito) y envía la trama al conmutador, que la entrega al portátil de Benito.

12. El portátil de Benito recibe la trama que contiene el mensaje de respuesta ARP y extrae la dirección MAC del router de pasarela (00:22:6B:45:1F:1B) de ese mensaje.
13. La computadora portátil de Benito podrá ahora (*¡finalmente!*) dirigir la trama Ethernet que contiene la consulta DNS hacia la dirección MAC del router de pasarela. Observe que el datagrama IP de esta trama tiene la dirección IP de destino 68.87.71.226 (el servidor DNS), mientras que la trama tiene la dirección de destino 00:22:6B:45:1F:1B (el router de pasarela). El portátil de Benito envía esta trama al conmutador, que la entrega al router de pasarela.

Seguimos con el inicio: enrutamiento dentro del dominio al servidor DNS

14. El router de pasarela recibe la trama y extrae el datagrama IP que contiene la consulta DNS. El router busca la dirección de destino de este datagrama (68.87.71.226) y determina a partir de su tabla de reenvío que el datagrama debe enviarse al router situado más a la izquierda dentro de la red de Comcast de la Figura 5.37. El datagrama IP se inserta dentro de una trama de la capa de enlace que resulte apropiada para el enlace que conecta el router de la facultad con el router de Comcast situado más a la izquierda, después de lo cual la trama se envía a través de ese enlace.
15. El router situado más a la izquierda dentro de la red de Comcast recibe la trama, extrae el datagrama IP, examina la dirección de destino del datagrama (68.87.71.226) y determina, gracias a su tabla de reenvío, la interfaz de salida a través de la cual debe reenviar el datagrama hacia el servidor DNS. La tabla de reenvío habrá sido previamente rellenada mediante el protocolo interno del dominio de Comcast (por ejemplo **RIP**, **OSPF** o **IS-IS**, Sección 4.6), así como mediante el **protocolo entre dominios de Internet**, **BGP**.
16. Finalmente, el datagrama IP que contiene la consulta DNS terminará por llegar al servidor DNS, el cual extrae el mensaje de consulta DNS, busca el nombre www.google.com en su base de datos DNS (Sección 2.5) y encuentra el **registro de recurso DNS** que contiene la dirección IP (64.233.169.105) para www.google.com (suponiendo que esa dirección esté actualmente almacenada en la caché del servidor DNS). Recuerde que estos datos de caché tienen su origen en el **servidor DNS autoritativo** (Sección 2.5.2) correspondiente a google.com. El servidor DNS compondrá un **mensaje de respuesta DNS** con la correspondencia entre el nombre de host y la dirección IP, después de lo cual inserta el mensaje de respuesta DNS en un segmento UDP e inserta este segmento en un datagrama IP dirigido a la computadora portátil de Benito (68.85.2.101). Este datagrama será reenviado de vuelta a través de la red de Comcast hasta el router de la facultad, y desde allí, a través del conmutador Ethernet a la computadora de Benito.
17. La computadora portátil de Benito extrae la dirección IP del servidor www.google.com del mensaje DNS. *Finalmente,* después de un *montón* de trabajo, la computadora portátil de Benito estará ya lista para contactar con el servidor www.google.com.

Interacción web cliente-servidor: TCP y HTTP

18. Ahora que el portátil de Benito dispone de la dirección IP de www. google.com, puede crear el **socket TCP** (Sección 2.7) que se utilizará para enviar el mensaje **GET HTTP** (Sección 2.2.3) a www.google.com. Cuando Benito crea el socket TCP, el protocolo TCP de su portátil tiene que llevar a cabo primero un proceso de **acuerdo en tres fases** (Sección 3.5.6) con el TCP de www.google.com. El portátil de Benito creará primero, por tanto, un segmento **SYN TCP** con puerto de destino 80 (para HTTP), insertará el segmento TCP dentro de un datagrama IP con una dirección de destino IP igual a 64.233.169.105 (www.google.com), incluirá el datagrama dentro de una trama con una dirección MAC de destino igual a 00:22:6B:45:1F:1B (el router de pasarela) y enviará la trama al conmutador.
19. Los routers de la red de la facultad, de la red de Comcast y de la red de Google reenvían el datagrama que contiene el segmento SYN TCP hacia www.google.com, utilizando la tabla de reenvío de cada router, como sucedía en los pasos 14–16 anteriores. Recuerde que las entradas de las tablas de reenvío de los routers que gobiernan el reenvío de paquetes a través del enlace entre dominios entre las redes de Comcast y de Google son determinadas mediante el protocolo **BGP** (Sección 4.6.3).
20. En algún momento, el datagrama que contiene el segmento SYN TCP llegará a www.google.com. El mensaje SYN TCP será extraído del datagrama y demultiplexado para ser entregado al socket de acogida asociado con el puerto 80. Se crea entonces un socket de conexión (Sección 2.7) para la conexión TCP entre el servidor HTTP de Google y la computadora portátil de Benito. Se genera entonces un segmento SYNACK TCP (Sección 3.5.6), se inserta dentro de un datagrama dirigido al portátil de Benito y, finalmente, se inserta dicho datagrama dentro de una trama de la capa de enlace que resulte apropiada para el enlace que conecta www.google.com con su router de primer salto.
21. El datagrama que contiene el segmento SYNACK TCP se reenvía a través de las redes de Google, de Comcast y de la facultad, terminando por llegar hasta la tarjeta Ethernet del portátil de Benito. El datagrama es demultiplexado dentro del sistema operativo y entregado al socket TCP creado en el Paso 18, con lo que entrará en estado conectado.
22. Ahora que el socket del portátil de Benito está (*¡finalmente!*) listo para enviar bytes a www.google.com, el navegador de Benito crea el mensaje GET HTTP (Sección 2.2.3) que contiene el URL que quiere extraer. Entonces, el mensaje GET HTTP se escribe en el socket, donde pasa a ser la carga útil de un segmento TCP. El segmento TCP se incluye en un datagrama y se envía y se entrega a www.google.com como en los Pasos 18–20.
23. El servidor HTTP en www.google.com lee el mensaje GET HTTP del socket TCP, crea un mensaje de **respuesta HTTP** (Sección 2.2), inserta el contenido de la página web solicitada en el cuerpo del mensaje de respuesta HTTP y envía el mensaje a través del socket TCP.
24. El datagrama que contiene el mensaje de respuesta HTTP se reenvía a través de las redes de Google, de Comcast y de la facultad y llega a la computadora portátil de Benito. El navegador web de Benito lee la respuesta HTTP del socket, extrae el código HTML correspondiente a la página web del cuerpo de la respuesta HTTP y, finalmente, (*¡finalmente!*) muestra la página web.

El escenario descrito cubre una gran cantidad de aspectos de la comunicación por red. Si ha comprendido la mayor parte del ejemplo anterior o todo él, entonces habrá avanzado

mucho desde que leyera por primera vez la Sección 1.1, donde decíamos que "buena parte de este libro trata de los protocolos de redes de computadoras", momento en el que posiblemente se estaba preguntando qué es un protocolo. Aunque el ejemplo anterior puede parecer bastante detallado, hemos omitido varios posibles protocolos adicionales (como por ejemplo, NAT ejecutándose en el router de pasarela de la facultad, el acceso inalámbrico a dicha red, los protocolos de seguridad para el acceso a red o para cifrar segmentos y datagramas, o los protocolos de gestión de red), así como diversas consideraciones adicionales (el almacenamiento en caché web, la jerarquía DNS) que podemos encontrar en la red Internet pública. Hablaremos de algunos de estos temas y otros en la segunda parte del libro.

Por último, cabe recalcar que el ejemplo anterior era una visión integrada y holística, aunque también refleja los componentes esenciales de muchos de los protocolos que hemos estudiado en esta primera parte del libro. El ejemplo pretendía centrarse más en el "cómo" que en el "por qué". Si desea una visión más amplia y reflexiva del diseño de los protocolos de red en general, consulte [Clark 1988, RFC 5218].

5.10 Resumen

En este capítulo hemos examinado la capa de enlace: sus servicios, los principios que subyacen a su funcionamiento y una serie de protocolos específicos importantes que utilizan esos principios a la hora de implementar servicios de la capa de enlace.

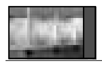
Hemos visto que el servicio básico de la capa de enlace consiste en mover un datagrama de la capa de red desde un nodo (router o host) hasta otro nodo adyacente. También hemos visto que todos los protocolos de la capa de enlace operan encapsulando un datagrama de la capa de red dentro de una trama de la capa de enlace antes de transmitir la trama a través del enlace existente hasta el nodo adyacente. Sin embargo, y yendo más allá de esta función común de entramado, hemos estudiado que los diferentes protocolos de la capa de enlace proporcionan servicios muy distintos de acceso al enlace, entrega (fiabilidad, detección/corrección de errores), control de flujo y transmisión (por ejemplo, *full-duplex* frente a semi-duplex). Estas diferencias se deben en parte a la amplia variedad de tipos de enlace sobre los que deben operar los protocolos de la capa de enlace. Un simple enlace punto a punto tiene un único emisor y un único receptor que se comunican a través de único "cable". Los enlaces de acceso múltiple, por su parte, son compartidos por varios emisores y receptores; en consecuencia, el protocolo de la capa de enlace para un canal de acceso múltiple dispone de un protocolo (su protocolo de acceso múltiple) para la coordinación del acceso al enlace. En el caso de MPLS, el "enlace" que conecta dos nodos adyacentes (por ejemplo, dos routers IP que sean adyacentes en sentido IP, es decir, que ambos son routers IP del siguiente salto hacia un determinado destino) puede ser en realidad una *red* en sí misma. En un cierto sentido, la idea de una red considerada como un enlace no debería resultar demasiado extraña. Un enlace telefónico que conecta una computadora/módem doméstico con un módem/router remoto, por ejemplo, es en realidad una ruta que pasa a través de una sofisticada y compleja *red* telefónica.

Entre los principios que subyacen a la comunicación de la capa de enlace, hemos examinado las técnicas de detección y corrección de errores, los protocolos de acceso múltiple, el direccionamiento de la capa de enlace, la virtualización (redes VLAN) y la construcción de redes LAN ampliadas mediante concentradores y conmutadores. En el caso de la detección y corrección de errores, hemos examinado cómo es posible añadir bits adicionales a la

cabecera de una trama con el fin de detectar, y en algunos casos de corregir, errores de inversión de bit que puedan producirse al transmitir la trama a través de un enlace. Hemos analizado los esquemas simples de paridad y de suma de comprobación, así como los códigos de redundancia cíclica más robustos. Después, hemos pasado a analizar el tema de los protocolos de acceso múltiple. Hemos identificado y estudiado tres enfoques generales para la coordinación del acceso a un canal de difusión: técnicas de particionamiento del canal (TDM, FDM), técnicas de acceso aleatorio (los protocolos ALOHA y CSMA) y técnicas de toma por turnos (sondeo y paso de testigo). Hemos visto que una consecuencia de hacer que múltiples nodos compartan un único canal de difusión era la necesidad de proporcionar direcciones de nodo en la capa de enlace. Hemos estudiado que las direcciones físicas son muy distintas de las direcciones de la capa de red y que, en el caso de Internet, se utiliza un protocolo especial (ARP, Protocolo de resolución de direcciones) para traducir entre estos dos tipos de direccionamiento. Después hemos examinado cómo los nodos que comparten un canal de difusión forman una red LAN y cómo pueden conectarse entre sí varias redes LAN para formar otras redes LAN de mayor tamaño, todo ello *sin* la intervención del enrutamiento de la capa red para interconectar esos nodos locales.

También hemos cubierto en detalle una serie de protocolos específicos de la capa de enlace: Ethernet y PPP. Hemos terminado nuestro estudio de la capa de enlace centrándonos en cómo las redes MPLS proporcionan servicios de la capa de enlace cuando inter-conectan routers IP. Hemos concluido el capítulo (y de hecho los primeros cinco capítulos) identificando los muchos protocolos necesarios para acceder a una simple página web. Habiendo cubierto la capa de enlace, *hemos concluido nuestro viaje descendente por la pila de protocolos.* Verdaderamente, la capa física se encuentra por debajo de la capa de enlace de datos, pero quizá sea mejor dejar los detalles de la capa física para otro curso (por ejemplo, un curso sobre teoría de la comunicación más que sobre redes de computadoras). No obstante, es cierto que hemos tocado varios aspectos de la capa física en este capítulo y en el Capítulo 1 (nuestra exposición acerca de los medios físicos de la Sección 1.2). Consideraremos de nuevo la capa física cuando estudiemos las características de los enlaces inalámbricos en el siguiente capítulo.

Aunque nuestro viaje por la pila de protocolos ya haya concluido, nuestro estudio de las redes de computadoras no ha terminado en modo alguno. En los siguientes cuatro capítulos nos ocuparemos de las redes inalámbricas, las redes multimedia, la seguridad de red y la gestión de red. Estos cuatro temas no encajan de manera natural en ninguna de las capas que conocemos. De hecho, cada uno de ellos cruza muchas de esas distintas capas. Comprender estos temas (que se califican de temas avanzados en algunos libros de texto) requiere por tanto un sólido conocimiento de todas las capas de la pila de protocolos, un conocimiento que nuestro estudio de la capa de enlace de datos nos ha permitido terminar de adquirir.



Problemas y cuestiones de repaso

Capítulo 5 Cuestiones de repaso

SECCIONES 5.1–5.2

R1. Considere la analogía de los transportes de la Sección 5.1.1. Si el pasajero es análogo a un datagrama, ¿qué sería análogo a la trama de la capa de enlace?

R2. Si todos los enlaces de Internet tuvieran que proporcionar un servicio de entrega fiable, ¿sería el servicio de entrega fiable de TCP redundante? ¿Por qué?

R3. ¿Cuáles son algunos de los posibles servicios que puede ofrecer un protocolo de la capa de enlace a la capa de red? ¿Cuáles de estos servicios de la capa de enlace tienen servicios correspondientes en IP? ¿Y en TCP?

SECCIÓN 5.3

R4. Suponga que dos nodos comienzan a transmitir al mismo tiempo un paquete de longitud L a través de un canal de difusión de velocidad R. Sea el retardo de propagación entre los dos nodos d_{prop}. ¿Se producirá una colisión si $d_{prop} < L/R$? ¿Por qué?

R5. En la Sección 5.3 hemos enumerado cuatro características deseables de un canal de difusión. ¿Cuáles de estas características presenta el protocolo ALOHA con particiones? ¿Cuáles de estas características presentan los protocolos de paso de testigo?

R6. Describa los protocolos de sondeo y de paso de testigo utilizando la analogía de las interacciones de las personas que asisten a un coctel.

R7. ¿Por qué el protocolo token-ring resulta ineficiente si una red LAN tiene un perímetro muy grande?

SECCIÓN 5.4

R8. ¿Cuál es el tamaño del espacio de direcciones MAC? ¿Y el del espacio de direcciones de IPv4? ¿Y el del espacio de direcciones de IPv6?

R9. Suponga que los nodos A, B y C están conectados a la misma red LAN de difusión (a través de sus adaptadores). Si A envía miles de datagramas IP a B, con cada trama que los encapsula dirigida hacia la dirección MAC de B, ¿procesará el adaptador de C estas tramas? En caso afirmativo, ¿pasará el adaptador de C los datagramas IP de dichas tramas a la capa de red de C? ¿Cómo variaría su respuesta si A envía las tramas con la dirección MAC de difusión?

R10. ¿Por qué las consultas ARP se envían dentro de una trama de difusión? ¿Por qué la respuesta ARP se envía dentro de una trama con una dirección MAC de destino específica?

R11. En la red de la Figura 5.19 el router tiene dos módulos ARP, cada uno con su propia tabla ARP. ¿Es posible que la misma dirección MAC aparezca en ambas tablas?

SECCIÓN 5.5

R12. Compare las estructuras de trama de Ethernet 10BASE-T, 100BASE-T y Gigabit. ¿En qué se diferencian?

R13. Suponga que un adaptador a 10 Mbps envía por un canal un flujo infinito de unos (1s) utilizando codificación Manchester. ¿Cuántas transiciones por segundo tiene la señal de salida del adaptador?

R14. En CSMA/CD, después de la quinta colisión, ¿cuál es la probabilidad de que un nodo seleccione $K = 4$? ¿A cuántos segundos de retardo corresponde el resultado $K = 4$ en una red Ethernet a 10 Mbps?

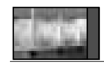
SECCIÓN 5.6

R15. Considere la Figura 5.26. ¿Cuántas subredes hay, en el sentido de direccionamiento explicado en la Sección 4.4?

SECCIÓN 5.7

R16. ¿Cuál es el número máximo de redes VLAN que pueden configurarse en un conmutador que soporta el protocolo 802.1Q? ¿Por qué?

R17. Suponga que tenemos que conectar N conmutadores que dan soporte a K grupos VLAN mediante un protocolo de enlace troncal (*trunking*)? ¿Cuántos puertos son necesarios para conectar los conmutadores? Justifique su respuesta.



Problemas

P1. Suponga que el contenido de información de un paquete es el patrón de bits 1110 1011 1001 1101 y que está utilizando un esquema de paridad par. ¿Cuál sería el valor del campo que contiene los bits de paridad para el caso de un esquema de paridad bidimensional? La respuesta debe ser tal que se utilice un campo de suma de comprobación de longitud mínima.

P2. Demuestre (proporcionando un ejemplo distinto del de la Figura 5.6) que los códigos de paridad bidimensional permiten corregir y detectar un único error de bit. Indique (proporcionando un ejemplo) un error doble de bit que pueda ser detectado pero no corregido.

P3. Suponga que la parte de información de un paquete (D en la Figura 5.4) contiene 10 bytes compuestos de la representación en código ASCII binario sin signo de 8 bits de la cadena de caracteres "Link Layer" (capa de enlace). Calcule la suma de comprobación de Internet para estos datos.

P4. Considere el problema anterior, pero suponga que los 10 bytes contienen:

a. la representación binaria de los números 1 a 10.

b. la representación ASCII de las letras A hasta J (mayúsculas).

c. la representación ASCII de las letras a hasta j (minúsculas).

Calcule la suma de comprobación de Internet para estos datos.

P5. Considere el generador de 7 bits, G = 10011, y suponga que D tiene el valor 1010101010. ¿Cuál es el valor de R?

P6. Considere el problema anterior, pero ahora suponga que D tiene el valor:

a. 1001000101.

b. 1010001111.

c. 0101010101.

P7. En este problema vamos a explorar algunas de las propiedades del código CRC. Para el generador G = 1001 dado en la Sección 5.2.3, responda a las siguientes cuestiones:

a. ¿Por qué puede detectar cualquier error simple de bit en los datos D?

b. ¿Puede el generador G anterior detectar cualquier número impar de errores de bit? ¿Por qué?

P8. En la Sección 5.3, hemos proporcionado un esbozo del cálculo de la eficiencia del protocolo ALOHA con particiones. En este problema vamos a completar dicho cálculo.

a. Recuerde que cuando hay N nodos activos, la eficiencia de ALOHA con particiones es $Np(1-p)^{N-1}$. Calcule el valor de p que maximiza esta expresión.

b. Utilizando el valor de p determinado en el apartado (a), calcule la eficiencia del protocolo ALOHA con particiones haciendo que N tienda a infinito. *Sugerencia*: $(1 - 1/N)^N$ tiende a $1/e$ cuando N tiende a infinito.

P9. Demuestre que la eficiencia máxima del protocolo ALOHA puro es $1/(2e)$. *Nota*: este problema es sencillo después de haber completado el problema anterior.

P10. Considere los nodos A y B que utilizan el protocolo ALOHA con particiones para competir por un canal. Suponga que el nodo A tiene más datos para transmitir que el nodo B, y que la probabilidad de retransmisión del nodo A, p_A, es mayor que la probabilidad de retransmisión del nodo B, p_B.

a. Proporcione una fórmula para la tasa media de transferencia del nodo A. ¿Cuál es la eficiencia total del protocolo con estos dos nodos?

b. Si $p_A = 2p_B$, ¿será la tasa media de transferencia de A el doble que la del nodo B? ¿Por qué? Si no es así, ¿cómo podemos seleccionar valores de p_A y p_B para que esto ocurra?

c. En general, suponga que hay N nodos, entre los que el nodo A tiene una probabilidad de retransmisión $2p$ y todos los demás nodos tienen una probabilidad de retransmisión p. Proporcione las expresiones necesarias para calcular las tasas medias de transferencia del nodo A y de los restantes nodos.

P11. Suponga que cuatro nodos activos (nodos A, B, C y D) están compitiendo por el acceso a un canal utilizando el protocolo ALOHA con particiones. Suponga que cada nodo tiene un número infinito de paquetes que transmitir y que cada nodo intenta transmitir en cada partición con una probabilidad p. La primera partición tiene el número 1, la segunda el número 2, etc.

a. ¿Cuál es la probabilidad de que el nodo A tenga éxito por primera vez en la partición 5?

b. ¿Cuál es la probabilidad de que algún nodo (A, B, C o D) tenga éxito en la partición 4?

c. ¿Cuál es la probabilidad de que el primer éxito suceda en la partición 3?

d. ¿Cuál es la eficiencia de este sistema de cuatro nodos?

P12. Dibuje una gráfica con la eficiencia de los protocolos ALOHA con particiones y ALOHA puro en función de p para los siguientes valores de N:

a. N 15.

b. N 20.

c. N 30.

P13. Considere un canal de difusión con N nodos y una tasa de transmisión de R bps. Suponga que el canal de difusión utiliza sondeo (con un nodo adicional de sondeo) para regular el acceso múltiple. Suponga que la cantidad de tiempo desde que un nodo completa una transmisión hasta que se le permite transmitir al nodo siguiente (es decir, el retardo de sondeo) es d_{sondeo}. Suponga que dentro de una ronda de sondeo, a cada nodo se le permite transmitir un máximo de Q bits. ¿Cuál es la tasa máxima de transferencia del canal de difusión?

P14. Considere tres redes LAN interconectadas mediante dos routers, como se muestra en la Figura 5.38.

a. Asigne direcciones IP a todas las interfaces. Para la Subred 1 utilice direcciones de la forma 192.168.1.xxx; para la Subred 2 utilice direcciones de la forma 192.168.2.xxx; y para la Subred 3 emplee direcciones de la forma 192.168.3.xxx.

b. Asigne direcciones MAC a todos los adaptadores.

c. Considere el envío de un datagrama IP desde el host E al host B. Suponga que todas las tablas ARP están actualizadas. Enumere todos los pasos, como hemos hecho en el ejemplo para un único router en la Sección 5.4.2.

d. Repita el apartado (c) suponiendo ahora que la tabla ARP del host emisor está vacía (y que todas las demás tablas están actualizadas).

P15. Considere la Figura 5.38. Ahora vamos sustituir el router situado entre las subredes 1 y 2 por un conmutador S1, y vamos a etiquetar el router situado entre las subredes 2 y 3 como R1.

a. Considere el envío de un datagrama IP desde el host E al host F. ¿Pedirá el host E al router R1 que le ayude a reenviar el datagrama? ¿Por qué? En la trama Ethernet

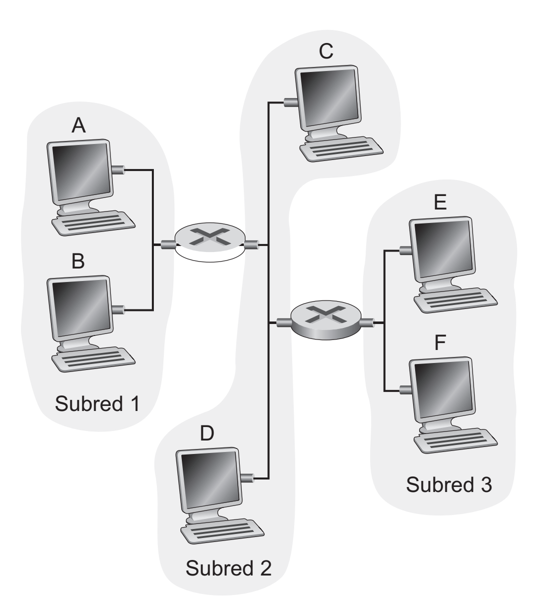


Figura 5.38 • Tres subredes interconectadas mediante routers.

que contiene el datagrama IP, ¿cuáles son las direcciones IP y MAC de origen y de destino?

b. Suponga que E quiere enviar un datagrama IP a B y suponga que la caché ARP de E no contiene la dirección MAC de B. ¿Realizará E una consulta ARP para averiguar la dirección MAC de B? ¿Por qué? En la trama Ethernet (que contiene el datagrama IP destinado a B) que se le entrega al router R1, ¿cuáles son las direcciones IP y MAC de origen y de destino?

c. Suponga que el host A quiere enviar un datagrama IP al host B, y que ni la caché ARP de A contiene la dirección MAC de B ni la caché ARP de B contiene la dirección MAC de A. Suponga además que la tabla de reenvío del conmutador S1 contiene entradas únicamente para el host B y el router R1. Por tanto, A difundirá un mensaje de solicitud ARP. ¿Qué acciones realizará el conmutador S1 una vez que reciba el mensaje de solicitud ARP? ¿Recibirá también el router R1 esta solicitud ARP? En caso afirmativo, ¿reenviará R1 el mensaje hacia la Subred 3? Una vez que el host B reciba este mensaje de solicitud ARP, devolverá al host A un mensaje de respuesta ARP. Pero ¿enviará un mensaje de consulta ARP para preguntar por la dirección MAC de A? ¿Por qué? ¿Que hará el conmutador S1 una vez que reciba el mensaje de respuesta ARP del host B?

P16. Considere el problema anterior, pero ahora suponga que el router situado entre las subredes 2 y 3 es sustituido por un conmutador. Responda a las preguntas de los apartados (a)–(c) del problema anterior en este nuevo contexto.

P17. Recuerde que con el protocolo CSMA/CD el adaptador espera K 512 periodos de bit después de producirse una colisión, donde K se selecciona aleatoriamente. Para K = 100, ¿cuánto tiempo espera el adaptador hasta volver al Paso 2 para una red Ethernet a 10 Mbps? ¿Y para una red Ethernet a 100 Mbps?

P18. Suponga que los nodos A y B se encuentran en el mismo bus Ethernet a 10 Mbps y que el retardo de propagación entre los dos nodos es igual a 325 periodos de bit. Suponga que el nodo A comienza a transmitir una trama y que, antes de finalizar, el nodo B comienza a transmitir otra trama. ¿Podría A finalizar la transmisión antes de detectar que B ha transmitido? ¿Por qué? Si la respuesta es afirmativa, entonces A creerá incorrectamente que su trama se ha transmitido con éxito sin que se haya producido una colisión. *Sugerencia*: suponga que en el instante $t = 0$ periodos de bit A comienza a transmitir una trama. En el caso peor, A transmitirá una trama de tamaño mínimo que ocupará 512 + 64 periodos de bit. Por lo que A terminaría de transmitir la trama para $t = 512 + 64$ periodos de bit. Por tanto, la respuesta es no si la señal de B alcanza a A antes del instante $t = 512 + 64$ periodos de bits. En el caso peor, ¿cuándo alcanzaría a A la señal de B?

P19. Explique por qué hace falta un tamaño mínimo de trama para Ethernet. Por ejemplo, Ethernet 10Base impone un tamaño mínimo de trama de 64 bytes (si ha hecho el problema anterior podría haberse dado cuenta de cuál es la razón). Ahora suponga que la distancia entre dos extremos de una red LAN Ethernet es d. ¿Puede hallar una fórmula que permite obtener el tamaño mínimo de trama necesario para los paquetes Ethernet? Basándose en su razonamiento, ¿cuál será el tamaño mínimo requerido de paquete de una red Ethernet que abarque 2 kilómetros?

P20. Suponga que quiere incrementar la velocidad de enlace de su cable Ethernet, ¿Cómo afectaría esta actualización a los tamaños mínimos de paquetes requeridos? Si actualiza su cable a una velocidad mayor y se da cuenta de que no puede cambiar el tamaño de paquete, ¿qué otra cosa puede hacer para que la red pueda seguir operando correctamente?

P21. Suponga que los nodos A y B se encuentran en el mismo bus Ethernet a 10 Mbps y que el retardo de propagación entre los dos nodos es igual a 245 periodos de bit. Suponga que A y B envían tramas al mismo tiempo, que las tramas colisionan y que luego A y B seleccionan diferentes valores de *K* en el algoritmo CSMA/CD. Suponiendo que no haya ningún otro nodo activo, ¿pueden colisionar las retransmisiones de A y B? Para nuestros propósitos, basta con resolver el siguiente ejemplo. Suponga que A y B comienzan la transmisión en $t = 0$ periodos de bit. Ambos detectan la colisión en el instante $t = 245$ periodos de bits y terminan de transmitir una señal de interferencia en $t = 245 + 48 = 293$ periodos de bit. Suponga que $K_A = 0$ y $K_B = 1$. ¿Para qué instante programará B su retransmisión? ¿En qué momento comenzará A su transmisión? (*Nota*: los nodos tienen que esperar a que el canal esté inactivo después de volver al Paso 2 (consulte el protocolo). ¿En qué momento alcanza a B la señal de A? ¿Se abstendrá B de transmitir en el instante programado?

P22. Considere una red Ethernet 100BASE-T a 100 Mbps con todos los nodos directamente conectados a un concentrador. Para tener una eficiencia de 0,50, ¿cuál debería ser la distancia máxima entre un nodo y el concentrador? Suponga una longitud de trama de 1.000 bytes y que no existe ningún repetidor. ¿Garantiza también esta distancia máxima que un nodo A que está transmitiendo será capaz de detectar si cualquier otro nodo ha transmitido mientras A estaba transmitiendo? ¿Por qué? ¿Cómo es esa distancia máxima, comparada con el estándar real a 100 Mbps? Suponga que la velocidad de propagación de la señal en Ethernet 100BASE-T es 1,8 10^8 m/s.

P23. Suponga que hay cuatro nodos, A, B, C y D, conectados a un concentrador mediante cables Ethernet a 10 Mbps. Las distancias entre el concentrador y estos cuatro nodos son 300, 400, 500 y 700 metros, respectivamente. Recuerde que para esta red Ethernet se utiliza el protocolo CSMA/CD. Suponga que la velocidad de propagación de la señal es 2 10^8 m/s.

a. ¿Cuál es la longitud mínima de trama requerida? ¿Cuál es la longitud máxima de trama requerida?
b. Si todas las tramas tiene una longitud de 1.500 bits, calcule la eficiencia de esta red Ethernet.

P24. En este problema tendrá que deducir la eficiencia de un protocolo de acceso múltiple de tipo CSMA/CD. En dicho protocolo, el tiempo está particionado y todos los adaptadores están sincronizados con las particiones. Sin embargo, a diferencia del protocolo ALOHA con particiones, la longitud de una partición (en segundos) es muy inferior al tiempo de trama (el tiempo necesario para transmitir una trama). Sea *S* la longitud de una partición. Suponga que todas las tramas tienen una longitud constante $L = kRS$, donde *R* es la velocidad de transmisión del canal y *k* es un entero de gran magnitud. Suponga que existen *N* nodos, cada uno con un número infinito de tramas para enviar. Suponga también que $d_{prop} < S$, de modo que todos los nodos pueden detectar una colisión antes de que finalice una partición de tiempo. El protocolo es como sigue:

- Si, para una partición determinada, ningún nodo está en posesión del canal, todos los nodos competirán por el acceso al canal; en particular, cada uno de los nodos transmite durante esa partición con una probabilidad p. Si exactamente un nodo transmite en la partición, dicho nodo se apropia del canal durante las subsiguientes $k - 1$ particiones y transmite su trama completa.
- Si algún nodo está en posesión del canal, todos los demás nodos se abstienen de transmitir hasta que el nodo que posee el canal ha terminado de transmitir su trama. Una vez que este nodo ha transmitido su trama, todos los nodos compiten por acceder al canal.

Observe que el canal alterna entre dos estados distintos: el estado productivo, que dura exactamente k particiones, y el estado no productivo, que dura un número aleatorio de particiones. Obviamente, la eficiencia del canal será la relación $k/(k + x)$, donde x es el número esperado de particiones no productivas consecutivas.

a. Para N y p fijos, determine la eficiencia de este protocolo.

b. Para un valor fijo N, determine el valor de p que maximiza la eficiencia.

c. Utilizando el valor de p (que es función de N) calculado en el apartado (b), determine la eficiencia cuando N tiende a infinito.

d. Demuestre que esta eficiencia tiende a 1 a medida que aumenta el tamaño de la trama.

P25. Suponga que dos nodos, A y B, están conectados en los extremos opuestos de un cable de 800 metros y que cada uno de ellos tiene una trama de 1.500 bits (incluyendo todas las cabeceras y preámbulos) que enviar al otro. Ambos nodos intentan transmitir en el instante $t = 0$. Suponga que existen cuatro repetidores entre A y B, cada uno de los cuales inserta un retardo de 20 bits. Suponga que la velocidad de transmisión es de 100 Mbps y que se utiliza CSMA/CD con intervalos de espera (*backoff*) que son múltiplos de 512 bits. Después de la primera colisión, A saca $K = 0$ y B saca $K = 1$ en el protocolo de *backoff* exponencial. Ignore la señal de interferencia y el retardo de 96 periodos de bit.

a. ¿Cuál es el retardo de propagación unidireccional (incluyendo los retardos de los repetidores) entre A y B en segundos? Suponga que la velocidad de propagación de la señal es igual a 2 10^8 m/s.

b. ¿En qué momento (en segundos) habrá sido entregado completamente a B el paquete de A?

c. Ahora suponga que sólo A tiene un paquete que enviar y que los repetidores se sustituyen por conmutadores. Suponga que cada conmutador tiene un retardo de procesamiento de 20 bits, además de un retardo de almacenamiento y reenvío. ¿En qué momento, en segundos, se entregará a B el paquete de A?

P26. En el estándar Ethernet, un emisor realiza una pausa de 96 periodos de bit entre el envío de dos tramas consecutivas. Este periodo de pausa se denomina hueco entre tramas y se utiliza para permitir que un dispositivo receptor complete el procesamiento de una trama recibida y se prepare para la recepción de la siguiente. Desde que el estándar Ethernet fuera especificado, se han producido importantes avances en la tecnología, incluyendo los relativos a la velocidad de los procesadores, de la memoria y de las pro-

pias velocidades de transmisión Ethernet. Si hubiera que reescribir el estándar, ¿cómo afectarían estas mejoras al hueco entre tramas?

P27. Considere la Figura 5.38 del Problema P14. Proporcione direcciones MAC e IP para las interfaces del host A, de ambos routers y del host F. Suponga que el host A envía un datagrama al host F. Indique las direcciones MAC de origen y de destino contenidas en la trama que encapsula este datagrama IP a medida que la trama es transmitida *(i)* de A al router de la izquierda, *(ii)* desde el router de la izquierda al router de la derecha y *(iii)* desde el router de la derecha al host F. Indique también las direcciones IP de origen y de destino del datagrama IP encapsulado dentro de la trama en cada uno de estos instantes de tiempo.

P28. Suponga ahora que el router de la izquierda de la Figura 5.38 se sustituye por un conmutador. Los hosts A, B, C y D y el router de la derecha se conectan en estrella a ese conmutador. Indique las direcciones MAC de origen y de destino contenidas en la trama que encapsula a este datagrama IP, a medida que la trama se transmite *(i)* desde A al conmutador, *(ii)* desde el conmutador al router de la derecha y *(iii)* desde el router de la derecha a F. Indique también las direcciones IP de origen y de destino contenidas en el datagrama IP encapsulado dentro de la trama en cada uno de estos instantes de tiempo.

P29. Considere la Figura 5.26. Suponga que todos los enlaces son a 100 Mbps. ¿Cuál es la tasa máxima total agregada de transferencia que puede conseguirse en los 9 hosts y los dos servidores de esta red? Puede suponer que cualquier host o servidor puede enviar a cualquier otro host o servidor. ¿Por qué?

P30. Suponga que los tres conmutadores departamentales de la Figura 5.26 se sustituyen por concentradores. Todos los enlaces son a 100 Mbps. Responda en estas condiciones a las cuestiones planteadas en el Problema P29.

P31. Suponga que *todos* los conmutadores de la Figura 5.26 son sustituidos por concentradores. Todos los enlaces son a 100 Mbps. Responda en estas condiciones a las cuestiones planteadas en el Problema P29.

P32. Considere la operación de un conmutador con aprendizaje en el contexto de la Figura 5.24. Suponga que *(i)* B envía una trama a E, *(ii)* E responde enviando una trama a B, *(iii)* A envía una trama a B, *(iv)* B responde enviando una trama a A. Inicialmente, la tabla del conmutador está vacía. Muestre el estado de la tabla del conmutador antes y después de cada uno de estos sucesos. Para cada suceso, identifique el enlace o los enlaces a través de los cuales se reenviará la trama transmitida y justifique brevemente sus respuestas.

P33. En este problema vamos a explorar el uso de pequeños paquetes para aplicaciones de Voz sobre IP. Uno de los inconvenientes de un tamaño de paquete pequeño es que una gran parte del ancho de banda del enlace es consumido por los bytes de sobrecarga. De cara al análisis, suponga que el paquete consta de P bytes de datos y 5 bytes de cabecera.

a. Considere el envío directo de un origen de voz codificado digitalmente. Suponga que el origen se codifica a una velocidad constante de 128 kbps. Suponga que cada paquete se rellena completamente antes de que el origen envíe el paquete a la red.

El tiempo requerido para rellenar un paquete se denomina **retardo de empaquetado**. En función de L, determine el retardo de empaquetado en milisegundos.

b. Los retardos de empaquetado superiores a 20 ms pueden dar lugar a un eco perceptible y desagradable. Determine el retardo de empaquetado para L = 1.500 bytes (lo que se corresponde aproximadamente con un paquete Ethernet de tamaño máximo) y para L = 50 (lo que se corresponde con un paquete ATM).

c. Calcule el retardo de almacenamiento y reenvío en un único conmutador para una velocidad de enlace de R = 622 Mbps para L = 1.500 bytes y para L = 50 bytes.

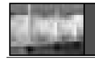
d. Comente las ventajas de utilizar un tamaño de paquete pequeño.

P34. Considere el único conmutador VLAN de la Figura 5.30 y suponga que se conecta un router externo al puerto 1 del conmutador. Asigne direcciones IP a los hosts de las redes IR y CC y a la interfaz del router. Indique los pasos seguidos tanto en la capa de red como en la capa de enlace para transferir un datagrama IP desde un host de IE a un host de CC (*Sugerencia:* vuelva a leer en el texto los comentarios acerca de la Figura 5.19).

P35. Considere la red MPLS mostrada en la Figura 5.36 y suponga que ahora los routers R5 y R6 son compatibles con MPLS. Suponga que deseamos realizar la ingeniería de tráfico de modo que los paquetes procedentes de R6 y destinados a A se conmuten hacia A a través de R6-R4-R3-R1, y que los paquetes procedentes de R5 destinados a A se conmuten a través de R5-R4-R2-R1. Detalle las tablas MPLS de R5 y R6, así como la tabla modificada de R4, que harían esto posible.

P36. Considere de nuevo el mismo escenario que el problema anterior, pero suponga que los paquetes de R6 destinados a D se conmutan a través de R6-R4-R3, mientras que los paquetes procedentes de R5 destinados a D se conmutan a través de R4-R2-R1-R3. Determine las tablas MPLS en todos los routers que harían esto posible.

P37. En este problema vamos a juntar muchas de las cosas que hemos aprendido acerca de los protocolos de Internet. Suponga que entra en una habitación, se conecta a Ethernet y desea descargar una página web. ¿Cuáles son todos los pasos de protocolo que tienen lugar, comenzando desde el instante en que enciende su PC y hasta el momento en que obtiene la página web? Suponga que no hay nada en la caché DNS ni en la caché del navegador cuando enciende su PC. (*Sugerencia:* los pasos incluyen el uso de los protocolos Ethernet, DHCP, ARP, DNS, TCP y HTTP.) Indique explícitamente en sus pasos cómo se obtienen las direcciones IP y MAC de un router de pasarela.



Preguntas para la discusión

Le animamos a explorar la Web a la hora de buscar las respuestas para las siguientes cuestiones.

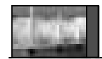
D1. Aproximadamente, ¿cuál es el rango de precios de un adaptador a 10/100 Mbps? ¿Y de un adaptador Ethernet Gigabit? ¿Cómo son estos precios, comparados con un módem de acceso telefónico a 56 kbps o con un módem ADSL?

D2. El precio de los conmutadores depende a menudo del número de interfaces (también denominadas *puertos* en la jerga de las redes LAN). Aproximadamente, ¿cuál es el rango de precio actual por interfaz para un conmutador que sólo dispone de interfaces a 100 Mbps?

D3. Muchas de las funciones de un adaptador pueden realizarse en un software que se ejecute en la CPU del nodo. ¿Cuáles son las ventajas y desventajas de pasar esta funcionalidad del adaptador al nodo?

D4. Busque en la Web los números de protocolo utilizados en una trama Ethernet para un datagrama IP y para un paquete ARP.

D5. Lea las referencias [Xiao 2000, Huang 2002 y RFC 3346] acerca de la ingeniería de tráfico mediante MPLS. Enumere una serie de objetivos de la ingeniería de tráfico. ¿Cuáles de estos objetivos sólo pueden satisfacerse utilizando MPLS y cuáles otros pueden satisfacerse utilizando protocolos existentes no MPLS? En el último caso, ¿qué ventajas ofrece MPLS?



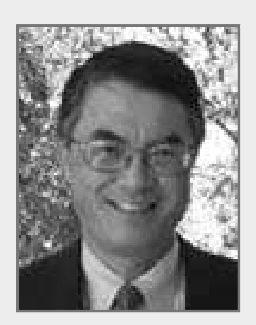
Prácticas de laboratorio con Wireshark

En el sitio web del libro, http://www.awl.com/kurose-ross, encontrará una práctica de laboratorio con Wireshark que permite examinar el funcionamiento del protocolo IEEE 802.3 y el formato de trama Wireshark.

Una segunda práctica de laboratorio con Wireshark examina las trazas de paquetes tomadas en un escenario de red doméstica similar al de la Figura 5.37.

ENTREVISTA CON...

Simon S. Lam



Simon S. Lam es catedrático y Regent Chair del Departamento de Ciencias de la Computación de la Universidad de Texas en Austin. Entre 1971 y 1974 estuvo trabajando en el Centro de Medidas de Red de ARPA en UCLA, donde trabajó con conmutación de paquetes vía satélite y vía radio. Dirigió un grupo de investigación que inventó los sockets seguros y prototipó, en 1993, la primera capa de sockets seguros (SSL) denominada *Secure Network Programming*, que ganó el premio al mejor sistema software de ACM en 2004. Sus intereses de investigación se encuentran en el campo del diseño y el análisis de protocolos de red y servicios de seguridad. Se graduó en la universidad del Estado de Washington, obteniendo su máster y su doctorado en UCLA. Fue elegido para la Academia Nacional de Ingeniería en 2007.

¿Por qué decidió especializarse en el campo de las redes?

Cuando llegué a UCLA como estudiante graduado en el otoño de 1969, mi intención era estudiar teoría de control. Entonces asisití a las clases de teoría de colas de Leonard Kleinrock y me quedé muy impresionado. Durante un tiempo estuve trabajando en el control adaptativo de sistemas de colas, como posible tema de mi tesis. A principios de 1972, Larry Roberts inició el proyecto Satellite System de ARPAnet (posteriormente denominado *Packet Satellite*). El profesor Kleinrock me pidió que me uniera al proyecto. Lo primero que hicimos fue añadir un algoritmo de *backoff* simple, aunque realista, al protocolo ALOHA con particiones. Poco después, me encontré con numerosos problemas de investigación interesantes, como el problema de inestabilidad de ALOHA y la necesidad de *backoff* adaptativo, temas que terminarían formando el núcleo de mi tesis.

Usted estuvo muy activo en los primeros días de Internet en la década de 1970, comenzando sus días de estudiante en UCLA. ¿Cómo eran las cosas entonces? ¿Alguien imaginaba lo que Internet llegaría a ser?

La atmósfera no era realmente distinta de la de otros proyectos de construcción de sistemas que yo había conocido en la industria y en las instituciones académicas. El objetivo inicialmente establecido para ARPAnet era bastante modesto, es decir, lo que se quería era proporcionar acceso a una serie de computadoras muy caras desde ubicaciones remotas, de modo que muchos más científicos pudieran utilizarlas. Sin embargo, con el inicio del proyecto *Packet Satellite* en 1972 y del proyecto *Packet Radio* en 1973, los objetivos de ARPA se habían ampliado sustancialmente. En 1973, ARPA estaba construyendo tres redes de paquetes distintas al mismo tiempo y Vint Cerf y Bob Kahn se vieron obligados a desarrollar una estrategia de interconexión.

En aquel entonces, todos estos desarrollos progresivos en el campo de las redes se veían (o eso creía) más que como algo lógico que como algo mágico. Nadie podría haber previsto la escala de Internet y la potencia actual de las computadoras personales. Pasó una década antes de que aparecieran los primeros PC. Para poner las cosas en perspectiva, la mayoría de los estudiantes enviaban sus programas de computadora en forma de pilas de fichas perforadas, para su procesamiento por lotes. Sólo algunos estudiantes tenían acceso directo a las computadoras, que normalmente estaban ubicadas en áreas restringidas. Los módems eran lentos y constituían todavía una rareza. Como estudiante gra-

duado lo único que yo tenía era un teléfono sobre mi mesa y utilizaba lápiz y papel para hacer la mayor parte de mi trabajo.

¿Hacia dónde cree que se encaminan el campo de la redes e Internet?

En el pasado, la simplicidad del protocolo IP de Internet era su mayor fortaleza a la hora de vencer otras soluciones competidoras y convertirse en el estándar *de facto* para la comunicación entre redes. A diferencia de otras soluciones competidoras, como X.25 en la década de 1980 y ATM en la década de 1990, IP puede ejecutarse por encima de cualquier tecnología de red de la capa de enlace, porque ofrece únicamente un servicio de datagramas de mejor esfuerzo. Por tanto, cualquier red de paquetes puede conectarse a Internet.

Hoy día, la mayor fortaleza de IP es en realidad una desventaja. IP es como una especie de camisa de fuerza que hace que los desarrollos de Internet queden constreñidos a una serie de direcciones específicas. En los últimos años muchos investigadores han redirigido sus esfuerzos, concentrándose únicamente en la capa de aplicación. También se están desarrollando numerosas investigaciones en el campo de las redes inalámbricas ad hoc, de las redes de sensores y de las redes por satélite. Estas redes pueden verse como sistemas autónomos o como sistemas de la capa de enlace, pudiendo florecer ese tipo de sistemas porque caen fuera de la camisa de fuerza representada por IP.

Muchas personas están entusiasmadas con las posibilidades que los sistemas P2P ofrecen como plataforma para el desarrollo de aplicaciones de Internet novedosas. Sin embargo, los sistemas P2P son muy ineficientes en su uso en los recursos de Internet. Una de las cosas que me preocupa es si la capacidad de transmisión y de conmutación del núcleo de Internet continuarán incrementándose más rápido que la demanda de tráfico a medida que Internet crezca para interconectar todo tipo de dispositivos y soportar las futuras aplicaciones de tipo P2P. Sin un sustancial sobredimensionamiento de la capacidad, garantizar la estabilidad de la red en presencia de ataques maliciosos y de congestión continuará representando un enorme desafío.

El enorme crecimiento de Internet también requiere asignar nuevas direcciones IP a gran velocidad a los operadores de red y a las empresas de todo el mundo. A la velocidad actual, el conjunto de direcciones IPv4 no asignadas se agotará en unos pocos años. Cuando eso suceda, sólo podrán asignarse grandes bloques contiguos del espacio de direcciones a partir del espacio de direcciones de IPv6. Como la adopción de IPv6 está siendo lenta en un principio, debido a la falta de incentivos para los nuevos usuarios, lo más probable es que IPv4 e IPv6 coexistan en Internet durante muchos años todavía. Una migración satisfactoria desde una Internet predominantemente IPv4 a otra predominantemente IPv6 requerirá un sustancial esfuerzo global.

¿Cuál es la parte más atractiva de su trabajo?

La parte más atractiva de mi trabajo como profesor es enseñar y motivar a *todos* los alumnos de mi clase y a *todos* los estudiantes de doctorado a los que superviso, en lugar de concentrarme sólo en los más brillantes. Las personas muy brillantes y motivadas pueden requerir alguna guía pero no mucho más. A menudo aprendo más de estos estudiantes de lo que ellos aprenden de mí. Lo más atractivo, el mayor desafío, es educar y motivar a los estudiantes que no son sobresalientes.

¿Qué impacto cree que tendrá la tecnología en el futuro sobre la enseñanza?

Antes o después, casi todos los conocimientos humanos serán accesibles a través de Internet, que será la herramienta más potente para el aprendizaje. Esta enorme base de conocimientos nos dará la posibilidad de nivelar el campo de juego para los estudiantes de todo el mundo. Por ejemplo, los estudiantes motivados en cualquier país podrán acceder a los sitios web de primera fila, a conferencias multimedia y a materiales formativos. Hoy día, se dice que las bibliotecas digitales de IEEE y ACM han acelerado el desarrollo de las investigaciones en ciencias de la computación en China. Con el tiempo, Internet eliminará todas las barreras geográficas en lo que a la enseñanza se refiere.



CAPÍTULO 6

Redes inalámbricas y móviles

En el mundo de la telefonía, los últimos 15 años han sido indudablemente la edad dorada de la telefonía celular. El número de abonados a los servicios de telefonía celular en todo el mundo ha pasado de 34 millones en 1993 a 4.000 millones a finales de 2008, sobrepasando ahora el número de abonados celulares al de líneas telefónicas normales [ITU Statistics 2009]. Las numerosas ventajas de los teléfonos celulares son evidentes para todo el mundo: disponemos de acceso a la red de telefonía global en cualquier momento, en cualquier lugar y sin ningún tipo de restricción, utilizando un dispositivo ligero y muy portátil. Con la llegada de las computadoras portátiles, las computadoras de mano, las PDA y su promesa de disponer de un acceso en cualquier momento, en cualquier lugar y sin ningún tipo de restricción a la Internet global, ¿nos estamos aproximando a una explosión similar en el uso de dispositivos Internet inalámbricos?

Independientemente del futuro crecimiento del uso de dispositivos Internet inalámbricos, ya está claro que las redes inalámbricas y los servicios relacionados con la movilidad que esas redes hacen posibles están aquí para quedarse entre nosotros. Desde el punto de vista de la comunicación por red, los desafíos planteados por estas redes, particularmente en las capas de enlace de datos y de red, son tan diferentes de los de las redes de computadoras cableadas tradicionales, que resulta imperativo dedicar todo un capítulo (es decir, *este* capítulo) al estudio de las redes inalámbricas y móviles.

Comenzaremos el capítulo con un estudio de los usuarios móviles, los enlaces inalámbricos y las redes inalámbricas, así como de su relación con las redes de mayor tamaño (normalmente cableadas) a las que se conectan. Estableceremos la distinción entre los desafíos planteados por la naturaleza *inalámbrica* de los enlaces de comunicaciones en dichas redes y por la *movilidad* que esos enlaces inalámbricos permiten. Al realizar esta importante distinción entre el carácter inalámbrico y la movilidad, podremos aislar, identificar y dominar

mucho mejor los conceptos clave de cada una de las áreas. Observe que existen, por supuesto, muchos entornos de red en los que los nodos de red son inalámbricos pero no móviles (por ejemplo, redes inalámbricas domésticas o empresariales con estaciones de trabajo estáticas y grandes pantallas de computadora), y que existen formas de movilidad que no requieren de enlaces inalámbricos (por ejemplo, un trabajador que utiliza una computadora portátil con conexión por cable en casa, apaga la computadora va hasta la oficina y conecta la computadora a la red cableada de la empresa). Por supuesto, muchos de los entornos de red más atractivos son aquellos en los que los usuarios son a la vez inalámbricos *y* móviles; por ejemplo, un escenario en el que un usuario móvil (situado por ejemplo en el asiento posterior de un vehículo) mantiene una llamada de voz sobre IP y múltiples conexiones TCP activas, mientras el vehículo circula por la autopista a 160 kilómetros por hora. Es aquí, en la intersección del carácter inalámbrico y la movilidad, donde encontraremos los desafíos técnicos más interesantes.

Comenzaremos ilustrando el entorno que vamos a emplear para nuestro análisis de la comunicación inalámbrica y la movilidad: una red en la que hay una serie de usuarios inalámbricos (y posiblemente móviles) conectados a otra infraestructura de red de mayor tamaño mediante un enlace inalámbrico situado en la frontera de la red. A continuación, pasaremos a estudiar las características de este enlace inalámbrico en la Sección 6.2. Incluimos una breve introducción al Acceso múltiple por división de código (CDMA, *Code Division Multiple Access*), un protocolo de acceso a un medio compartido que se emplea a menudo en las redes inalámbricas. Dicho estudio lo haremos también en la Sección 6.2. En la Sección 6.3, examinaremos con un cierto grado de profundidad los aspectos de nivel de enlace del estándar de red LAN inalámbrica IEEE 802.11 (WiFi); también dedicaremos unas pocas palabras a Bluetooth y WiMAX. En la Sección 6.4 veremos una panorámica del acceso celular a Internet, incluyendo las emergentes tecnologías celulares 3G que proporcionan tanto voz como acceso a Internet a alta velocidad. En la Sección 6.5 fijaremos nuestra atención en la movilidad, centrándonos en los problemas de localizar a un usuario móvil, efectuar el enrutamiento hasta ese usuario móvil e ir "transfiriendo" al usuario móvil, que se está desplazando dinámicamente desde un punto de conexión a la red hasta otro. Examinaremos cómo se implementan estos servicios móviles en el estándar de IP móvil y en GSM en las Secciones 6.6 y 6.7, respectivamente. Por último, en la Sección 6.8, consideraremos el impacto de los enlaces inalámbricos y la movilidad sobre los protocolos de la capa de transporte y las aplicaciones en red.

6.1 Introducción

La Figura 6.1 muestra el escenario con el que vamos a analizar los temas de la comunicación inalámbrica de datos y la movilidad. Comenzaremos manteniendo nuestro estudio en un nivel lo suficientemente general como para cubrir un amplio rango de redes, incluyendo tanto las redes LAN inalámbricas (como por ejemplo IEEE 802.11) y las redes celulares (como una red 3G); en secciones posteriores profundizaremos en un análisis más detallado de determinadas arquitecturas inalámbricas más específicas. Dentro de una red inalámbrica podemos identificar los siguientes elementos:

- *Hosts inalámbricos.* Como en el caso de las redes cableadas, los hosts son los dispositivos que actúan como sistemas terminales y que ejecutan las aplicaciones. Un **host ina-**

HISTORIA

ACCESO WIFI PÚBLICO: ¿ESTARÁ PRONTO DISPONIBLE EN LOS SEMÁFOROS?

Los puntos de acceso WiFi (WiFi *hotspots*, ubicaciones públicas en las que los usuarios pueden encontrar acceso inalámbrico 802.11) están siendo cada vez más comunes en los hoteles, aeropuertos y cafés de todo el mundo.

A finales de 2008, T-Mobile proporciona accesos de este tipo en más de 10.000 ubicaciones de Estados Unidos (y en más de 45.000 en todo el mundo), incluyendo las cafeterías Starbucks y las tiendas Borders Books & Music. La mayoría de los campus universitarios ofrecen un acceso inalámbrico ubicuo y resulta difícil encontrar un hotel en el que no exista acceso inalámbrico a Internet. Muchas ciudades, entre las que se incluyen Filadelfia, San Francisco, Toronto y Hong Kong han anunciado planes para proporcionar acceso inalámbrico ubicuo dentro de la ciudad. El objetivo en Filadelfia era "convertir Filadelfia en el área de acceso WiFi más grande de Estados Unidos y ayudar a mejorar la educación, eliminar las diferencias sociales de carácter digital, mejorar el desarrollo de los barrios y reducir el coste de administración". Inicialmente el plan requería instalar puntos de acceso inalámbrico 802.11b en, aproximadamente, 4.000 semáforos y dispositivos de control de tráfico. El ambicioso programa, un acuerdo entre la ciudad, Wireless Philadelphia (una entidad sin ánimo de lucro) y Earthlink (un proveedor de servicios Internet), ha llegado a construir una red operativa. El proyecto GovWiFi de Hong Kong "proporcionará servicio Wi-Fi gratuito en 350 oficinas gubernamentales en una serie de fases. Se situarán unos 2.000 puntos de acceso Wi-Fi públicos dentro del territorio y se hará de Hong Kong una ciudad inalámbrica, con cerca de 10.000 puntos de acceso Wi-Fi públicos en 2009".

Sin embargo, llevar a cabo planes para un acceso WiFi ubicuo a nivel municipal presenta ciertas dificultades. La red WiFi municipal de San Francisco nunca ha llegado a pasar de la fase de propuesta. En 2008, Earthlink dió por finalizado su servicio WiFi en Filadelfia. Pero la red WiFi municipal que opera en el centro de Toronto y que no es gratuita continúa estando operativa, así como las redes WiFi municipales de otras varias ciudades y pueblos más pequeños. Los intentos de disponer de redes inalámbricas municipales continúan. http://www.muniwireless.com/ es un sitio web en el que se realiza un seguimiento del panorama, siempre cambiante, de las redes inalámbricas municipales.

lámbrico puede ser una computadora portátil, una computadora de mano, una PDA, un teléfono o una computadora de escritorio. Los hosts en sí pueden ser móviles o no.

- *Enlaces inalámbricos.* Un host se conecta a una estación base (definida más adelante) o a otro host inalámbrico a través de un **enlace de comunicaciones inalámbrico**. Las diferentes tecnologías de enlace inalámbrico tienen distintas velocidades de transmisión y pueden transmitir a diferentes distancias. La Figura 6.2 muestra dos características clave (área de cobertura y velocidad del enlace) de los estándares más populares de redes inalámbricas. (La figura sólo pretende proporcionar una idea aproximada de estas características. Por ejemplo, algunos de estos tipos de redes sólo ahora se están comenzado a implantar y algunas velocidades de enlace pueden aumentar o disminuir respecto a los valores mostrados, dependiendo de la distancia, de las condiciones del canal y del número de usuarios en la red inalámbrica.) Nos ocuparemos de estos estándares más ade-

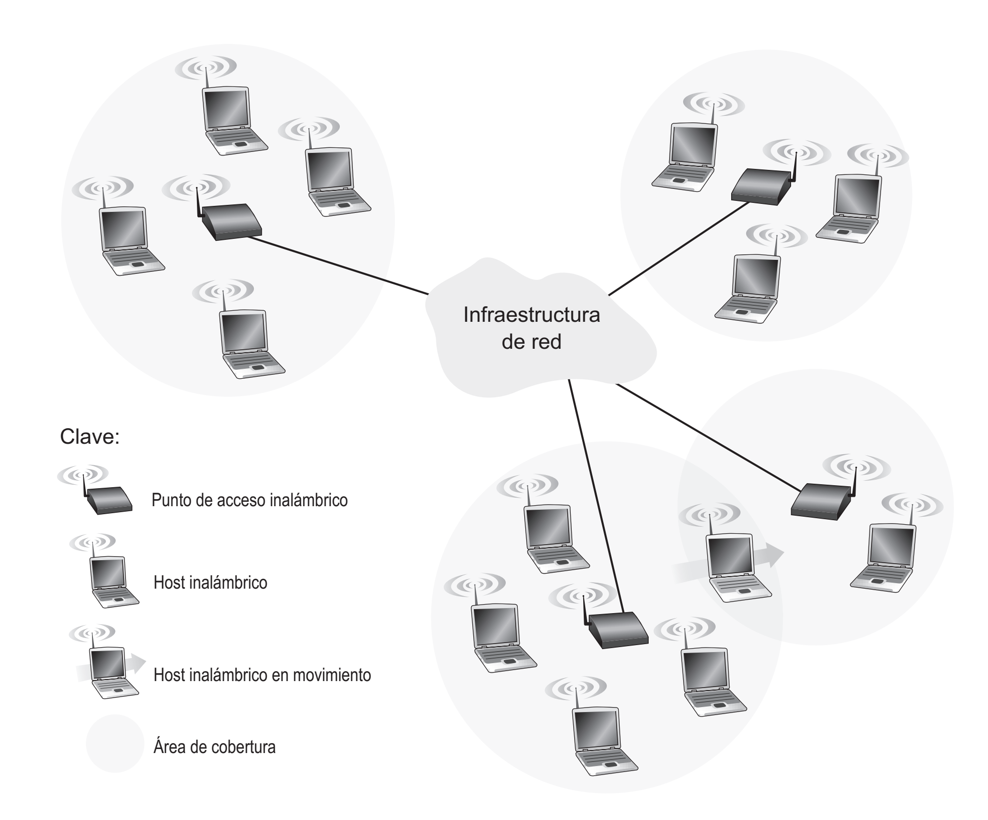


Figura 6.1 • Elementos de una red inalámbrica.

lante a lo largo de la primera mitad de este capítulo; también consideraremos otras características de los enlaces inalámbricos, como sus tasas de error de bit y las causas de esos errores, en la Sección 6.2.

En la Figura 6.1 una serie de enlaces inalámbricos conectan a un conjunto de hosts inalámbricos ubicados en la frontera de la red con la infraestructura de esa red de mayor tamaño. Conviene añadir que los enlaces inalámbricos también se utilizan en ocasiones *dentro* de una red para conectar entre sí routers, conmutadores y otros equipos de red. Sin embargo, nuestro enfoque en este capítulo se centrará en el uso de las comunicaciones inalámbricas alrededor de las fronteras de la red, ya que es ahí donde podemos encontrar los desafíos técnicos más atractivos y donde se está experimentado un auténtico crecimiento.

- *Estación base.* La **estación base** es una parte clave de la infraestructura de la red inalámbrica. A diferencia del host inalámbrico y de los enlaces inalámbricos, una estación base no tiene un equivalente obvio dentro de las redes cableadas. La estación base es responsable de enviar y recibir datos (es decir, paquetes) hacia y desde un host inalámbrico que esté asociado con esa estación base. La estación base será a menudo responsable de coordinar la transmisión de los múltiples hosts inalámbricos que estén asociados con ella. Cuando decimos que un host inalámbrico está "asociado" con una estación base, queremos decir que (1) el host se encuentra dentro de la distancia máxima de comunicación

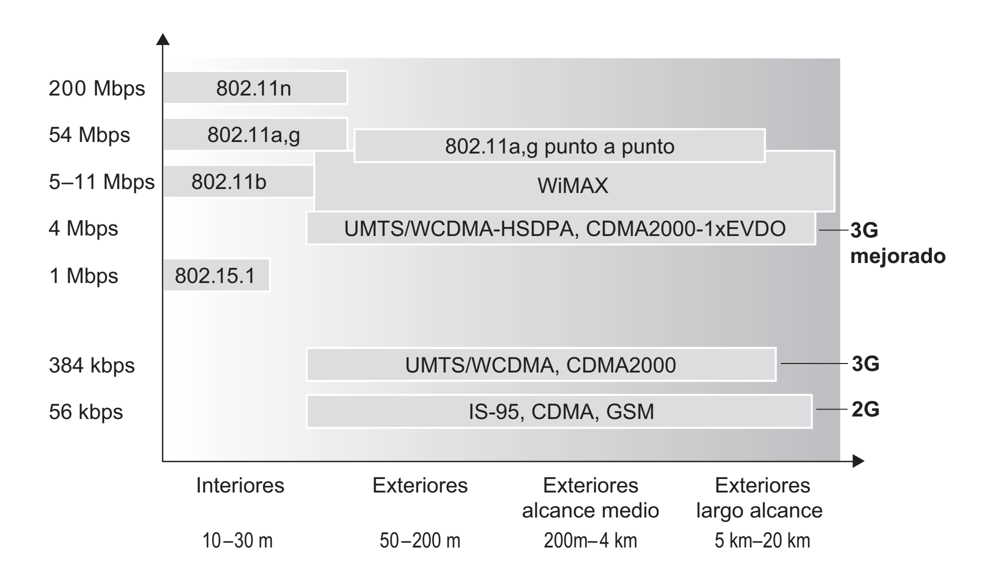


Figura 6.2 • Características del enlace para una serie de estándares seleccionados de redes inalámbricas.

inalámbrica de la estación base y (2) el host utiliza la estación base para reenviar datos hacia y desde la red de mayor tamaño. Las **torres de telefonía** en las redes celulares y los **puntos de acceso** en las redes LAN inalámbricas 802.11 son ejemplos de estaciones base.

En la Figura 6.1, la estación base está conectada a la red de mayor tamaño (por ejemplo, a Internet, a una red doméstica o corporativa o a una red telefónica), funcionando así como retransmisor de la capa de enlace entre el host inalámbrico y el resto del mundo con el que el host se comunica.

De los hosts asociados con una estación base se suele decir que operan en **modo de infraestructura**, puesto que todos los servicios de red tradicionales (como por ejemplo, la asignación de direcciones y el enrutamiento) son proporcionados por la red con la que un host se conecta a través de la estación base. En las **redes ad hoc**, los hosts inalámbricos no tienen ninguna infraestructura de ese tipo a la que conectarse. En ausencia de dicha infraestructura, los propios hosts tienen que proporcionar servicios tales como el enrutamiento, la asignación de direcciones, la traducción de nombres de tipo DNS, etc.

Cuando un host móvil se desplaza fuera del alcance de una estación base y entra dentro del área de cobertura de otra, cambia su punto de conexión con la red de mayor tamaño (es decir, cambia la estación base con la que está asociado); este proceso se conoce con el nombre de **transferencia** (*handoff*). Este tipo de movilidad plantea numerosos y complejos problemas. Si un host puede moverse, ¿cómo podemos averiguar la ubicación actual del host móvil dentro de la red, para poder reenviar datos a ese host móvil? ¿Cómo se lleva a cabo el direccionamiento, sabiendo que un host puede estar en una de muchas posibles ubicaciones? Si el host se mueve *durante* una conexión TCP o llamada telefónica, ¿cómo se pueden enrutar los datos, para que la conexión continúe activa de forma

ininterrumpida? Éstas y otras muchas (¡muchísimas!) cuestiones hacen de las redes inalámbricas y móviles un área de investigación particularmente atractiva.

- *Infraestructura de red.* Ésta es la red de mayor tamaño con la que un host inalámbrico puede querer comunicarse.

Habiendo examinado los elementos de una red inalámbrica, observemos que estos elementos pueden combinarse de muchas formas distintas para componer distintos tipos de redes inalámbricas. Conocer una taxonomía de estos tipos de redes inalámbricas puede ser útil a la hora de leer este capítulo o a la hora de leer o aprender más acerca de las redes inalámbricas más allá de lo que en este libro se expone. En el nivel más general, podemos clasificar las redes inalámbricas según dos criterios: *(i)* si un paquete dentro de la red inalámbrica realiza exactamente *un salto inalámbrico o varios saltos inalámbricos* y *(ii)* si existe una *infraestructura*, como por ejemplo una estación base, dentro de la red:

- *Redes basadas en infraestructura y un único salto.* Estas redes tienen una estación base conectada a una red cableada de mayor tamaño (por ejemplo, Internet). Además, toda la comunicación se realiza entre esta estación base y un host inalámbrico, con un único salto inalámbrico. Las redes 802.11 que utilizamos en las aulas, en las cafeterías o en las bibliotecas; las redes de telefonía celular y las redes 802.16 WiMAX, de las que pronto hablaremos, caen todas ellas dentro de esta categoría.
- *Redes sin infraestructura y un único salto.* En estas redes no existe una estación base conectada a una red inalámbrica. Sin embargo, como veremos, uno de los nodos de esta red de un único salto puede coordinar las transmisiones de los restantes nodos. Las redes Bluetooth (que estudiaremos en la Sección 6.3.6) y las redes 802.11 en modo ad hoc son redes sin infraestructura y de un único salto.
- *Redes basadas en infraestructura y múltiples saltos.* En estas redes existe una estación base que está cableada a la red de mayor tamaño. Sin embargo, algunos nodos inalámbricos pueden tener que retransmitir sus comunicaciones a través de otros nodos inalámbricos con el fin de comunicarse a través de la estación base. Algunas redes de sensores inalámbricos y las denominadas **redes de malla inalámbricas** caen dentro de esta categoría.
- *Redes sin infraestructura y múltiples saltos.* En estas redes no existe una estación base y los nodos pueden tener que retransmitir sus mensajes a través de otros diversos nodos para alcanzar un cierto destino. Los nodos también pueden ser móviles, con lo que la conectividad entre los nodos irá variando, lo que constituye una clase de redes conocidas con el el nombre de **redes móviles ad hoc (MANET, *Mobile ad hoc network*)**. Si los nodos móviles son vehículos, la red se denomina **red vehicular ad hoc (VANET, *Vehicular Ad hoc NETwork*)**. Como puede imaginar, el desarrollo de protocolos para tales redes es enormemente complicado y es materia de muchas investigaciones que están actualmente en marcha.

En este capítulo vamos a limitarnos fundamentalmente a la redes de un único salto y, dentro de ellas, a las redes basadas en infraestructura.

Pero examinemos ahora con mayor profundidad los desafíos técnicos que surgen dentro de las redes inalámbricas y móviles. Comenzaremos considerando los enlaces inalámbricos individuales y dejando nuestro estudio acerca de las cuestiones de movilidad para más adelante dentro del capítulo.

6.2 Características de las redes y enlaces inalámbricos

Comencemos considerando una simple red cableada, por ejemplo una red doméstica, con un conmutador Ethernet cableado (véase la Sección 5.6) que interconecta los hosts. Si reemplazamos la Ethernet cableada por una red inalámbrica 802.11, tendríamos que sustituir las tarjetas Ethernet cableadas de los hosts por tarjetas NIC inalámbricas y cambiar el conmutador Ethernet por un punto de acceso inalámbrico, pero no haría falta prácticamente ningún cambio en la capa de red ni en las capas superiores. Esto sugiere que debemos centrar nuestra atención en la capa de enlace a la hora de buscar diferencias importantes entre las redes cableadas e inalámbricas. De hecho, podemos encontrar varias distinciones de importancia entre un enlace cableado y un enlace inalámbrico:

- *Intensidad decreciente de la señal*. La radiación electromagnética se atenúa a medida que va atravesando la materia (por ejemplo, una señal de radio que atraviesa una pared). Incluso en el espacio vacío la señal se dispersará, lo que da como resultado una intensidad de señal decreciente (en ocasiones denominada **pérdida de propagación***, path loss*) a medida que se incrementa la distancia entre el emisor y el receptor.
- *Interferencias de otros orígenes.* Los orígenes de radio que transmiten en la misma banda de frecuencia interferirán entre sí. Por ejemplo, los teléfonos inalámbricos a 2,4 GHz y las redes LAN inalámbricas 802.11b transmiten en la misma banda de frecuencias. Por tanto, el usuario de una red LAN inalámbrica 802.11b que esté hablando a través de un teléfono inalámbrico a 2,4 GHz puede esperar que ni la red ni el teléfono tengan un comportamiento particularmente satisfactorio. Además de las interferencias de los orígenes de transmisión, el ruido electromagnético presente en el entorno (por ejemplo, un motor cercano o un microondas) también pueden provocar interferencias.
- *Propagación multicamino.* La **propagación multicamino** (*multipath*) tiene lugar cuando partes de la onda electromagnética se reflejan en los objetos y en el suelo, tomando caminos de diferentes longitudes entre un emisor y un receptor. Esto hace que la señal recibida sea menos limpia en el receptor. El desplazamiento de objetos situados entre el emisor y el receptor puede hacer que la propagación multicamino varíe a lo largo del tiempo.

Para ver una explicación detallada de las características, modelos y medidas de los canales inalámbricos, consulte [Anderson 1995].

La exposición anterior sugiere que los errores de bit serán más comunes en los enlaces inalámbricos que en los enlaces cableados. Por esta razón, no resulta sorprendente que los protocolos de enlace inalámbrico (como el protocolo 802.11 que examinaremos en la siguiente sección) no sólo empleen potentes códigos CRC para la detección de errores, sino también protocolos de la capa de enlace con transferencia de datos fiable que se encargan de retransmitir las tramas corrompidas.

Habiendo considerado los problemas que pueden afectar a un canal inalámbrico, volvamos nuestra atención al host que recibe la señal inalámbrica. El host recibe una señal electromagnética que es una combinación de una forma degradada de la señal original transmitida por el emisor (degradada debido a los efectos de la atenuación y de la propagación multicamino que hemos visto anteriormente, entre otros) y del ruido de fondo presente en el entorno. La **relación señal-ruido (SNR, *Signal-to-Noise Ratio*)** es una medida relativa de la intensidad de la señal recibida (es decir, de la información que se está transmi-

tiendo) y de este ruido. Normalmente, la relación señal-ruido se mide en unidades de decibelios (dB), que es una unidad de medida que algunos creen que los ingenieros eléctricos utilizan principalmente para confundir a los informáticos. La SNR, medida en dB, es veinte veces la relación del logaritmo en base 10 de la amplitud de la señal recibida y la amplitud del ruido. De cara a lo que aquí nos ocupa, lo único que necesitamos saber es que, cuanto mayor sea la relación SNR, más fácil le será al receptor extraer la señal transmitida del ruido de fondo.

La Figura 6.3 (adaptada de [Holland 2001]) muestra la tasa de errores de bit (BER, *Bit Error Rate*) que, dicho de forma simple, es la probabilidad de que un bit transmitido llegue de forma errónea al receptor, en función de la SNR para tres técnicas de modulación distintas utilizadas para codificar la información para su transmisión a través de un canal inalámbrico idealizado. La teoría de la modulación y de la codificación, así como las técnicas de extracción de la señal y de análisis de BER, caen fuera del alcance de este libro (consulte [Schwartz 1980] si desea obtener más información sobre estos temas). De todos modos, la Figura 6.3 ilustra varias características de la capa física que son importantes a la hora de comprender los protocolos de comunicación inalámbrica de las capas superiores.

- *Para un determinado esquema de modulación, cuanto mayor es la SNR menor es la BER.* Dado que un emisor puede incrementar la SNR aumentando su potencia de transmisión, podrá reducir la probabilidad de que una trama se reciba de forma errónea aumentando esa potencia de transmisión. Observe, sin embargo, que existe muy poca ventaja práctica cuando se incrementa la potencia más allá de un cierto umbral para, por ejemplo, reducir la tasa de errores de bit (BER) de 10^{-12} a 10^{-13}. También existen *desventajas* asociadas con ese incremento de la potencia de transmisión: el emisor tendrá que gastar más energía (lo que es una consideración de gran importancia para los usuarios móviles alimentados por baterías) y es más probable que las transmisiones del emisor interfieran con las de otros emisores (véase la Figura 6.4(b)).

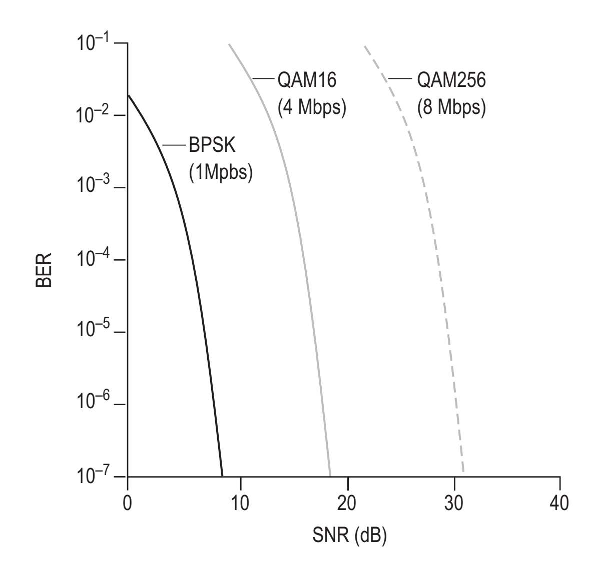


Figura 6.3 • Tasa de errores de bit, velocidad de transmisión y SNR.

- *Para una SNR dada, una técnica de modulación con una velocidad de transmisión de bit más alta (independientemente de si esos bits son erróneos o no) tendrá una tasa de errores de bit mayor.* Por ejemplo, en la Figura 6.3, con una SNR de 10 dB, la modulación BPSK con una velocidad de transmisión de 1 Mbps tiene una BER inferior a 10^{-7}, mientras que con la modulación QAM16, con una velocidad de transmisión de 4 Mbps, la BER es 10^{-1}, demasiado alta como para resultar útil en la práctica. Sin embargo, con una SNR de 20 dB, la modulación QAM16 tiene una velocidad de transmisión de 4 Mbps y una BER de 10^{-7}, mientras que la modulación BPSK tiene una velocidad de transmisión de sólo 1 Mbps y una BER que es demasiado baja y cae (literalmente) “fuera del gráfico”. Si podemos tolerar una BER de 10^{-7}, la mayor velocidad de transmisión ofrecida por QAM16 haría que fuera la técnica de modulación preferida para esa situación. Estas consideraciones nos conducen a la característica final que describimos a continuación.
- *Puede utilizarse una selección dinámica de la técnica de modulación de la capa física para adaptar la técnica de modulación a las condiciones del canal.* La SNR (y por tanto la BER) puede variar como resultado de la movilidad o debido a cambios en el entorno. En los sistemas celulares de transmisión de datos y en las redes 802.16 WiMAX y 802.11 WiFi, que examinaremos en la Sección 6.3, se usan técnicas de codificación y modulación adaptativas. Esto permite, por ejemplo, la selección de una técnica de modulación que proporcione la máxima velocidad de transmisión, sujeta a una restricción relativa a la BER, para unas determinadas características del canal.

Una tasa de errores de bit más alta y variable en el tiempo no es la única diferencia entre un enlace cableado y un enlace inalámbrico. Recuerde que en el caso de los enlaces cableados de difusión, todos los nodos reciben las transmisiones realizadas por los restantes nodos. En el caso de los enlaces inalámbricos, la situación no es tan simple como se muestra en la Figura 6.4. Suponga que la estación A está transmitiendo hacia la estación B. Suponga también que la estación C está transmitiendo hacia la estación B. Con el denominado **problema del terminal oculto**, las obstrucciones físicas presentes en el entorno (por ejemplo, una montaña o un edificio) pueden impedir que A y C escuchen las transmisiones del otro, incluso aún cuando las transmisiones de A y C estén interfiriéndose mutuamente en el des-

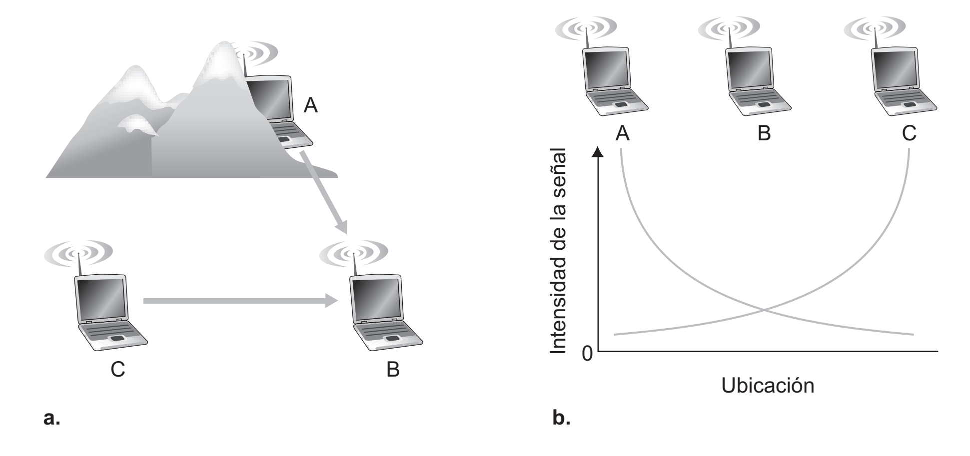


Figura 6.4 • Problema del terminal oculto causado por la existencia de un obstáculo (a) y problema del desvanecimiento (b).

tino B. Esto se muestra en la Figura 6.4(a). Un segundo escenario que da como resultado la presencia de colisiones indetectables en el receptor es el debido al **desvanecimiento** de la intensidad de la señal a medida que ésta se propaga a través del medio inalámbrico. La Figura 6.4(b) ilustra el caso en el que A y C están colocadas de tal forma que sus señales no son lo suficientemente intensas como para que puedan ambas estaciones detectar las transmisiones de la otra, a pesar de lo cual esas señales *tienen* una intensidad suficiente como para interferir entre sí en la estación B. Como veremos en la Sección 6.3, los problemas del terminal oculto y del desvanecimiento hacen que el acceso múltiple en una red inalámbrica sea considerablemente más complejo que en una red cableada.

6.2.1 CDMA

Recuerde del Capítulo 5 que cuando los hosts se comunican a través de un medio compartido, se necesita un protocolo para que las señales enviadas por varios emisores no interfieran en los receptores. En el Capítulo 5 hemos descrito tres clases de protocolos de acceso al medio: particionamiento del canal, acceso aleatorio y toma de turnos. El Acceso múltiple por división de código (CDMA, *Code Division Multiple Access*) pertenece a la familia de protocolos de particionamiento del canal. Es el protocolo prevalente en las tecnologías celulares y de redes LAN inalámbricas. Puesto que CDMA es tan importante en el mundo inalámbrico, vamos a echar ahora un rápido vistazo a CDMA antes de entrar en las secciones posteriores a analizar tecnologías específicas del acceso inalámbrico.

En un protocolo CDMA, cada bit enviado se codifica multiplicándolo por una señal (el código) que varía a una velocidad mucho mayor (conocida con el nombre de **velocidad de chip**, *chipping rate*) que la secuencia original de bits de datos. La Figura 6.5 muestra un escenario idealizado de codificación/decodificación CDMA. Suponga que definimos la unidad de tiempo según la velocidad a la que llegan al codificador CDMA los bits de datos originales; es decir, cada bit original de datos que haya que transmitir requiere una partición de tiempo de un bit. Sea d_i el valor del bit de datos para la *i-ésima* partición de bit. Por comodidad matemática, vamos a representar los bits de datos que tengan un valor 0 como –1. Cada partición de bit se subdivide a su vez en M mini-particiones; en la Figura 6.5 $M = 8$, aunque en la práctica M es mucho mayor. El código CDMA utilizado por el emisor está compuesto por una serie de M valores, c_m, $m = 1, \ldots, M$, cada uno de los cuales tiene el valor +1 o –1. En el ejemplo de la Figura 6.5, el código CDMA de M bits que está utilizando el emisor es (1, 1, 1, –1, 1, –1, –1, –1).

Para ilustrar cómo funciona CDMA, vamos a centrarnos en el *i-ésimo* bit de datos, d_i. Para la *m-ésima* mini-partición del tiempo de transmisión de bit de d_i, la salida del codificador CDMA, $Z_{i,m}$, es el valor de d_i multiplicado por el *m-ésimo* bit del código asignado CDMA, c_m:

$$Z_{i,m} = d_i \quad c_m \tag{6.1}$$

En un mundo ideal en el que no existieran otros emisores interfiriendo, el receptor recibiría los bits codificados, $Z_{i,m}$, y recuperaría el bit de datos original, d_i, realizando el cálculo:

$$d_i = \frac{1}{M} \sum_{m=1}^{M} Z_{i,m} \quad c_m \tag{6.2}$$

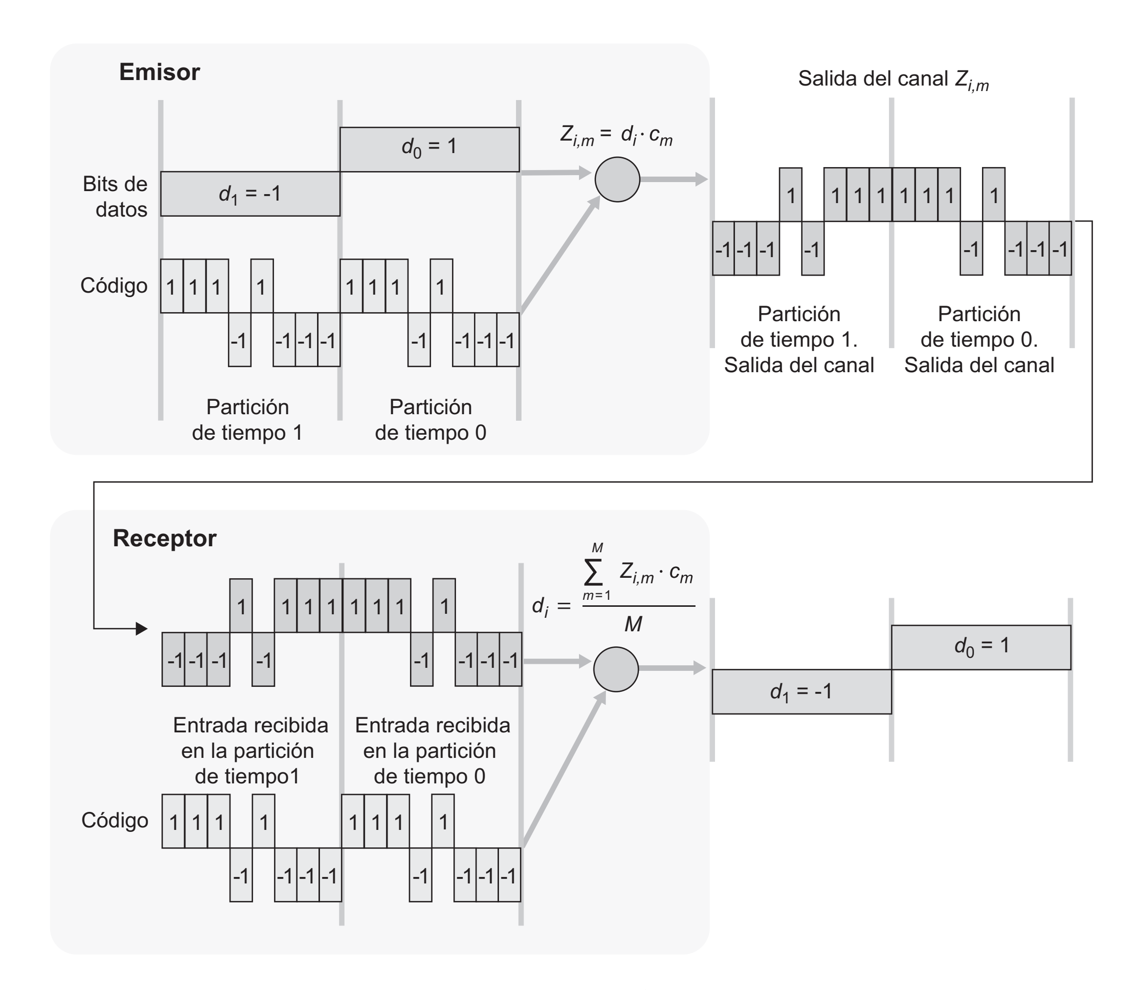


Figura 6.5 • Un ejemplo simple de CDMA: codificación del emisor y decodificación del receptor.

El lector puede trabajar los detalles del ejemplo de la Figura 6.5 para ver que los bits de datos originales se recuperan, efectivamente, de modo correcto en el receptor utilizando la Ecuación 6.2.

Sin embargo, el mundo dista mucho de ser ideal y, como hemos dicho anteriormente, CDMA debe trabajar en presencia de otros emisores que interfieren y que están codificando y transmitiendo sus datos utilizando otro código asignado diferente. Pero, ¿cómo puede un receptor CDMA recuperar los bits de datos originales del emisor cuando esos bits de datos están entremezclados con los bits transmitidos por otros emisores? CDMA funciona bajo la suposición de que las señales interferentes de los bits transmitidos son aditivas. Esto significa, por ejemplo, que si los tres emisores envían un valor 1 y un cuarto emisor envía un valor –1 durante la misma mini-partición, entonces la señal recibida en todos los receptores durante esa mini-partición será un 2 (dado que $1 + 1 + 1 - 1 = 2$). En presencia de múltiples emisores, el emisor *s* calcula sus transmisiones codificadas, $Z^s_{i,m}$, exactamente de la misma forma que en la Ecuación 6.1. El valor que llega a un receptor durante la *m-ésima*

mini-partición de la *i-ésima* partición de bit será ahora, sin embargo, la *suma* de los bits transmitidos por los N emisores durante dicha mini-partición:

$$Z^*_{i,m} = \sum_{s=1}^{N} Z^s_{i,m}$$

Sorprendentemente, si se eligen cuidadosamente los códigos de los emisores, cada receptor puede recuperar los datos enviados por un determinado emisor a partir de la señal agregada, simplemente utilizando el código del emisor exactamente en la misma forma que en la Ecuación 6.2:

$$d_i = \frac{1}{M} \sum_{m=1}^{M} Z^*_{i,m} \quad c_m \tag{6.3}$$

como se muestra en la Figura 6.6 para un ejemplo de CDMA con dos emisores. El código CDMA de M bits utilizado por el emisor de la parte superior es (1, 1, 1, –1, 1, –1, –1, –1), mientras que el código CDMA empleado por el emisor de la parte inferior es (1, –1, 1, 1, 1, –1, 1, 1). La Figura 6.6 ilustra el caso de un receptor que recupera los bits de datos originales correspondientes al emisor de la parte superior. Observe que el receptor es capaz de extraer los datos del emisor 1 a pesar de que estén siendo interferidos por la transmisión correspondiente al emisor 2.

Recuerde la analogía del coctel del Capítulo 5. Un protocolo CDMA es similar al caso en que los participantes en la reunión hablen en múltiples idiomas; en tales circunstancias, los seres humanos somos bastante buenos a la hora de centrarnos en la conversación que se está manteniendo en el idioma que comprendemos, al mismo tiempo que filtramos las restantes conversaciones. Podemos ver, a partir de estas explicaciones, que CDMA es un protocolo de particionamiento, en el sentido de que particiona el espacio de códigos (en lugar del tiempo o la frecuencia) y asigna a cada nodo una parte dedicada de ese espacio de códigos.

El análisis que aquí hemos realizado de CDMA es necesariamente breve; en la práctica, es preciso contemplar diversas cuestiones relativamente complicadas. En primer lugar, para que los receptores CDMA sean capaces de extraer una señal de un emisor concreto, los códigos CDMA deben elegirse cuidadosamente. En segundo lugar, nuestro análisis partía de la suposición de que las intensidades de las señales recibidas de los diversos emisores eran iguales, pero en la realidad esto puede ser difícil de conseguir. Existe bastante literatura técnica en la que se analizan estas y otras cuestiones relacionadas con CDMA; consulte [Pickholtz 1982; Viterbi 1995] para ver más detalles.

6.3 WiFi: redes LAN inalámbricas 802.11

Presentes por todas partes, en las oficinas, en los domicilios particulares, en las instituciones educativas, en la cafeterías, en los aeropuertos e incluso en la esquina de cualquier calle, las redes LAN inalámbricas son hoy en día una de las tecnologías más importantes de redes de acceso para Internet. Aunque en la década de 1990 se desarrollaron muchas tecnologías y estándares para redes LAN inalámbricas, hay una clase concreta de estándares que ha terminado por emerger como ganador indiscutible: la red **LAN inalámbrica IEEE 802.11**, tam-

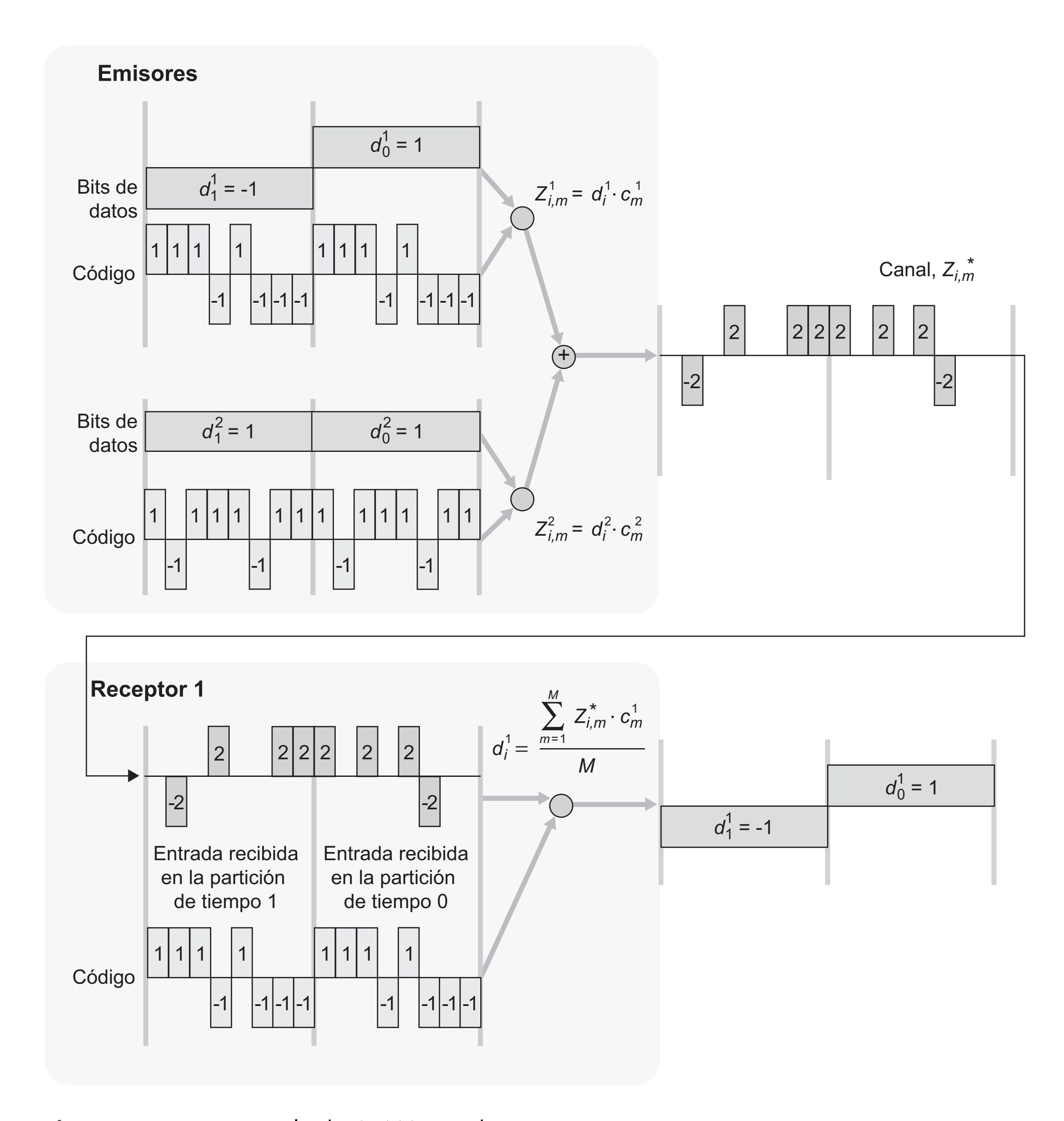


Figura 6.6 • Un ejemplo de CDMA con dos emisores.

bién conocida como red **WiFi**. En esta sección vamos a examinar en detalle las redes LAN inalámbricas 802.11, su estructura de trama, su protocolo de acceso al medio y la interconexión de las redes LAN 802.11 con las redes LAN Ethernet cableadas.

Existen varios estándares 802.11 para la tecnología LAN inalámbrica, incluyendo 802.11b, 802.11a y 802.11g. La Tabla 6.1 resume las características principales de estos estándares. En el momento de escribir este libro (primavera de 2009), hay muchos más

Estándar	Rango de frecuencias (Estados Unidos)	Velocidad de datos
802.11b	2,4–2,485 GHz	hasta 11 Mbps
802.11a	5,1–5,8 GHz	hasta 54 Mbps
802.11g	2,4–2,485 GHz	hasta 54 Mbps

Tabla 6.1 • Resumen de los estándares IEEE 802.11.

dispositivos 802.11g que están siendo ofrecidos por los fabricantes de puntos de acceso y tarjetas LAN. También hay disponibles diversos dispositivos en modo dual (802.11a/g) y trimodo (802.11a/b/g).

Los tres estándares 802.11 comparten muchas características. Todos ellos emplean el mismo protocolo de acceso al medio, CSMA/CA, del que pronto hablaremos. Los tres usan también la misma estructura de trama para la capa de enlace. Los tres estándares tienen la capacidad de reducir su velocidad de transmisión para poder alcanzar mayores distancias. Y los tres estándares permiten trabajar tanto "en modo de infraestructura" como en "modo ad hoc", como pronto veremos. Sin embargo, como se muestra en la Tabla 6.1, los tres estándares presentan algunas diferencias importantes en la capa física.

La tecnología LAN inalámbrica 802.11b ofrece una velocidad de datos de 11 Mbps y opera en la banda de frecuencias sin licencia de 2,4–2,485 GHz, compitiendo por el espectro de frecuencias con los teléfonos a 2,4 GHz y los hornos microondas. Las redes LAN inalámbricas 802.1la pueden operar a velocidades de bit significativamente mayores, pero lo hacen a frecuencia más altas. Operando a una mayor frecuencia, las redes LAN 802.11a tienen una distancia de transmisión más corta para un determinado nivel de potencia y se ven más afectadas por los problemas de la propagación multicamino. Las redes LAN 802.11g operan en la misma banda de menor frecuencia que el estándar 802.11b y son compatibles hacia atrás con 802.11b (de modo que, en una red, los clientes 802.11b se pueden actualizar incrementalmente), a pesar de lo cual utilizan las velocidades de transmisión mayores del estándar 802.11a, lo que permite a los usuarios disfrutar de lo mejor los otros dos estándares.

Hay un nuevo estándar WiFi, 802.11n [IEEE 802.11n 2009], que se encuentra en proceso de estandarización. 802.11n utiliza antenas de entrada múltiple y salida múltiple (MIMO, *Multiple-Input Multiple-Output*); es decir, dos o más antenas en el lado emisor y dos o más antenas en el lado receptor están transmitiendo/recibiendo diferentes señales [Diggavi 2004]. Aunque todavía el proceso de estandarización no ha concluido, ya hay disponibles productos pre-estándar, en los que las primeras pruebas realizadas muestran que se pueden conseguir en la práctica tasas de transferencia de transmisión de más de 200 Mbps [Newman 2008]. Un problema importante del actual borrador del estándar es la forma en que los dispositivos 802.11n interactuarán con los dispositivos 802.11a/b/g existentes.

6.3.1 La arquitectura 802.11

La Figura 6.7 ilustra los principales componentes de la arquitectura de una red LAN inalámbrica 802.11. El componente fundamental de la arquitectura 802.11 es el **conjunto de servicio básico (BSS, *Basic Service Set*)**. Un BSS contiene una o más estaciones inalámbricas y

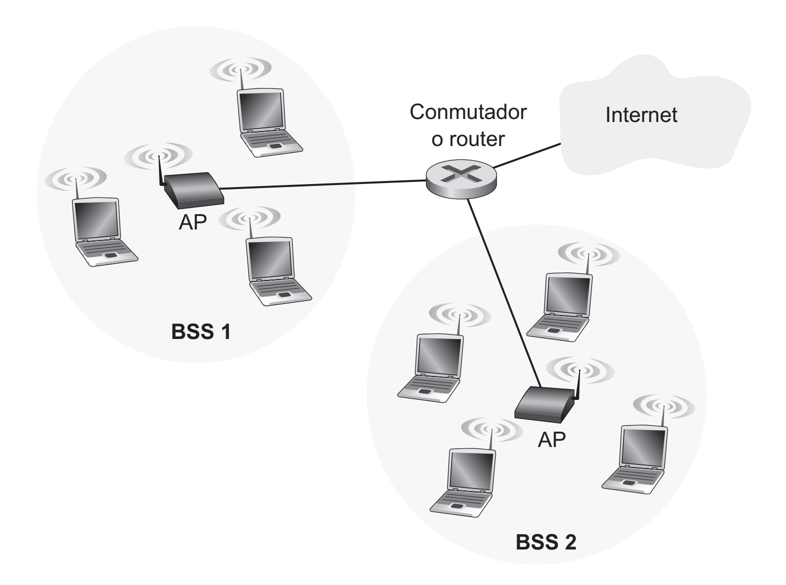


Figura 6.7 • Arquitectura de una red LAN IEEE 802.11.

una **estación base** central, conocida con el nombre de **punto de acceso (AP, *Access Point*)** en terminología 802.11. La Figura 6.7 muestra el punto de acceso en cada uno de los dos BSS; los puntos de acceso se interconectan a un dispositivo de interconexión (como un conmutador o un router), que a su vez lleva hacia Internet. En una red doméstica típica existirá un punto de acceso y un router (normalmente integrados en una misma unidad), que conectarán el BSS con Internet.

Al igual que sucede con los dispositivos Ethernet, cada estación inalámbrica 802.11 tiene una dirección MAC de 6 bytes que está almacenada en el firmware de la tarjeta adaptadora de la estación (es decir, en la tarjeta de interfaz de red 802.11). Cada punto de acceso tiene también una dirección MAC para su interfaz inalámbrica. Al igual que sucede con Ethernet, estas direcciones MAC son administradas por el IEEE y son (en teoría) globalmente únicas.

Como hemos dicho en la Sección 6.1, las redes LAN inalámbricas que incorporan puntos de acceso suelen denominarse **redes LAN inalámbricas de infraestructura**, siendo la “infraestructura” los puntos de acceso junto con la infraestructura de Ethernet cableada que interconecta los puntos de acceso y un router. La Figura 6.8 muestra que las estaciones IEEE 802.11 también pueden agruparse para formar una red ad hoc: una red sin ningún control central y que no tiene conexiones con el “mundo exterior”. En este caso, la red es formada “sobre la marcha” por una serie de dispositivos móviles que se han encontrado con que están próximos entre sí, que tienen una necesidad de comunicarse y que no encuentran ninguna infraestructura de red preexistente en la ubicación en la que están. Una red ad hoc podría formarse cuando una serie de personas con computadoras portátiles se juntan (por ejemplo, en una sala de conferencias, en un tren o en un vehículo) y quieren intercambiar datos en ausencia de un punto acceso centralizado. Se ha generado un enorme interés en las redes ad hoc, ya que los dispositivos portátiles capaces de comunicarse entre sí continúan proliferando. En esta sección, sin embargo, centraremos nuestra atención en las redes LAN inalámbricas de infraestructura.

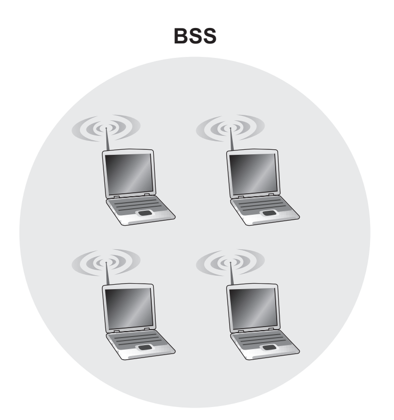


Figura 6.8 • Una red ad hoc IEEE 802.11.

Canales y asociación

En la arquitectura 802.11 cada estación inalámbrica necesita asociarse con un punto de acceso antes de poder enviar o recibir datos de la capa de red. Aunque todos los estándares 802.11 utilizan el mecanismo de asociación, analizaremos el tema específicamente en el contexto de IEEE 802.11b/g.

Cuando un administrador de red instala un punto de acceso, asigna un **Identificador de conjunto de servicio (SSID, *Service Set Identifier*)** de una o dos palabras a ese punto de acceso. (Por ejemplo, cuando se "ven las redes disponibles" en Microsoft Windows XP, aparece una lista que muestra el identificador SSID de cada punto de acceso que esté dentro del alcance.) El administrador debe también asignar un número de canal a ese punto de acceso. Para comprender los números de canal, recuerde que 802.11 opera en el rango de frecuencias de 2,4 GHz a 2,485 GHz. Dentro de esta banda de 85 MHz, 802.11 define 11 canales parcialmente solapados. Dados dos canales cualesquiera diremos que no se solapan si y sólo si están separados por cuatro o más canales. En particular, el conjunto de canales 1, 6 y 11 es el único conjunto de tres canales no solapados. Esto quiere decir que un administrador podría crear una red LAN inalámbrica con una velocidad máxima de transmisión agregada de 33 Mbps instalando tres puntos de acceso 802.11b en la misma ubicación física, asignando los canales 1, 6 y 11 a los puntos de acceso e interconectando todos los puntos de acceso mediante un conmutador.

Ahora que tenemos unos conocimientos básicos de los canales 802.11, vamos a describir una situación interesante (y que no resulta tan rara): la de la jungla WiFi. Una **jungla WiFi** es cualquier ubicación física en la que una estación inalámbrica está recibiendo una señal suficientemente intensa desde dos o más puntos de acceso. Por ejemplo, en muchas cafeterías de la ciudad de Nueva York una estación inalámbrica puede captar la señal de numerosos puntos de acceso cercanos. Uno de los puntos de acceso puede ser gestionado por la propia cafetería, mientras que los otros pueden encontrarse en viviendas situadas cerca de la misma. Cada uno de estos puntos de acceso estará, probablemente, ubicado en una subred IP diferente y se le habrá asignado un canal de manera independiente.

Suponga ahora que entramos en dicha jungla WiFi con nuestra computadora portátil, buscando poder acceder a Internet de manera inalámbrica. Suponga que existen cinco puntos de acceso en esa jungla WiFi. Para poder obtener acceso a Internet, nuestra estación

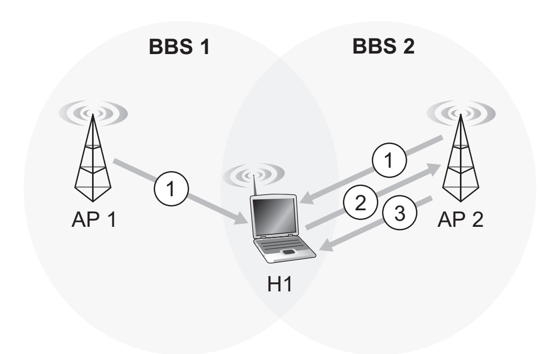
inalámbrica necesita unirse a exactamente una de las subredes y, por tanto, necesitará **asociarse** con exactamente uno de los puntos de acceso. Asociarse quiere decir que la estación inalámbrica creará un cable virtual entre ella misma y el punto de acceso. Específicamente, sólo el punto de acceso asociado enviará tramas de datos (es decir, tramas que contienen datos, como por ejemplo un datagrama) a nuestra estación inalámbrica y nuestra estación inalámbrica enviará tramas de datos hacia Internet solamente a través del punto de acceso asociado. Pero, ¿cómo se asocia una estación inalámbrica con un punto de acceso concreto? Todavía más importante: ¿cómo sabe una estación inalámbrica qué puntos de acceso hay en esa jungla, si es que hay alguno?

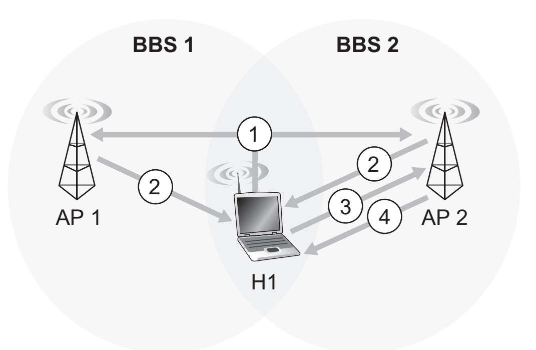
El estándar 802.11 requiere que un punto de acceso envíe de forma periódica **tramas baliza** (*beacon frames*), cada una de las cuales incluye la dirección MAC y el identificador SSID del punto de acceso. La estación inalámbrica, que sabe que los puntos de acceso están enviando tramas baliza, explora los once canales buscando la tramas baliza de cualquier punto de acceso que pueda haber en las proximidades (algunos de los cuales pueden estar transmitiendo a través del mismo canal, ya que estamos en una jungla). Habiendo determinado qué puntos de acceso hay disponibles a través de las tramas baliza, seleccionamos (o nuestro host inalámbrico selecciona) uno de los puntos de acceso para llevar a cabo la asociación.

El estándar 802.11 no especifica un algoritmo para seleccionar con cuál de los puntos de acceso disponibles asociarse; dicho algoritmo se deja al arbitrio de los diseñadores del software y el firmware 802.11 del host inalámbrico. Normalmente, el host elige el punto de acceso cuya trama baliza se recibe con la máxima intensidad de señal. Pero, aunque una alta intensidad de señal resulta conveniente (vea, por ejemplo, la Figura 6.3), la intensidad de la señal no es la única característica del punto de acceso que influirá en el rendimiento que un host perciba. En particular, es posible que el punto de acceso seleccionado pueda tener una gran intensidad de señal, pero que pueda estar sobrecargado por otra serie de hosts asociados (que necesitarán compartir el ancho de banda inalámbrico disponible en dicho punto de acceso), mientras que se deja sin seleccionar un punto de acceso bastante descargado, debido a que la intensidad de la señal es ligeramente menor. Por esto, recientemente se han propuesto diversas formas de elección de los puntos de acceso [Vasudevan 2005; Nicholson 2006; Sudaresan 2006]. Para ver una interesante discusión práctica acerca de cómo se mide la intensidad de la señal, consulte [Bardwell 2004].

El proceso de exploración de los canales y de escucha de las tramas baliza se conoce con el nombre de **exploración pasiva** (véase la Figura 6.9a). Un host inalámbrico también puede realizar una **exploración activa**, difundiendo una trama de sondeo que será recibida por todos los puntos de acceso que caigan dentro del alcance del host inalámbrico, como se muestra en la Figura 6.9b. Los puntos de acceso responden a la trama de la solicitud de sondeo con una trama de respuesta de sondeo. El host inalámbrico puede entonces elegir el punto de acceso con el que asociarse de entre todos aquellos que hayan respondido.

Después de seleccionar el AP con el que asociarse, el host inalámbrico envía una trama de solicitud de asociación a ese punto de acceso, el cual responde con una trama de respuesta de asociación. Observe que este segundo acuerdo de solicitud/respuesta también es necesario cuando se utiliza la exploración activa, dado que un punto de acceso que está respondiendo a la trama inicial de solicitud de sondeo no sabe cuál de los (posiblemente numerosos) puntos de acceso que hayan respondido va a seleccionar el host para asociarse, de la misma manera que un cliente DHCP puede seleccionar entre múltiples servidores DHCP (véase la Figura 4.21). Una vez asociado con un punto de acceso, el host se unirá a la





a. Exploración pasiva
1. Tramas baliza enviadas desde los puntos de acceso.
2. Envío de la trama de solicitud de asociación desde H1 al AP seleccionado.
3. Envío de la trama de respuesta de asociación desde el AP seleccionado a H1.

b. Exploración activa
1. Difusión desde H1 de una trama de solicitud de sondeo.
2. Envío de tramas de respuesta de sondeo desde los AP.
3. Envío de trama de solicitud de asociación desde H1 al AP seleccionado.
4. Envío de trama de respuesta de asociación desde el AP seleccionado a H1

Figura 6.9 • Exploración activa y pasiva de los puntos de acceso.

subred (en el sentido de direccionamiento IP de la Sección 4.4.2) a la que pertenezca el punto de acceso. Normalmente el host enviará un mensaje de descubrimiento DHCP (véase la Figura 4.21) hacia la subred a través del punto de acceso para obtener una dirección IP de esa subred. Una vez obtenida la dirección, el resto del mundo verá entonces a dicho host simplemente como otro host cualquiera con una dirección IP perteneciente a dicha subred.

Para poder crear una asociación con un punto de acceso concreto, puede que la estación inalámbrica tenga que autenticarse ante el punto de acceso. Las redes LAN inalámbricas 802.11 proporcionan diversas alternativas para la autenticación y el acceso. Una técnica, utilizada por muchas compañías, permite el acceso a una red inalámbrica basándose en la dirección MAC de la estación. Una segunda técnica, empleada en muchos cafés Internet, utiliza nombres de usuario y contraseñas. En ambos casos, el punto de acceso se comunica normalmente con un servidor de autenticación, reenviando la información intercambiada entre la estación terminal inalámbrica y el servidor de autenticación utilizando un protocolo como RADIUS [RFC 2865] o DIAMETER [RFC 3588]. Separar el servidor de autenticación del punto de acceso permite que un único servidor de autenticación proporcione servicio a muchos puntos de acceso distintos, centralizando las (a menudo sensibles) decisiones relativas a la autenticación y al acceso en ese único servidor y manteniendo los costes y la complejidad de los puntos de acceso relativamente bajos. Veremos en la Sección 8.8 que el nuevo protocolo IEEE 802.11i, que define los aspectos de seguridad de la familia de protocolos 802.11, adopta precisamente este enfoque.

6.3.2 El protocolo MAC 802.11

Una vez asociada una estación inalámbrica con un punto de acceso, la estación puede comenzar a enviar y recibir tramas de datos hacia y desde el punto de acceso. Pero, dado que puede haber múltiples estaciones que pueden desear transmitir tramas de datos al mismo tiempo a través del mismo canal, es preciso utilizar un protocolo de acceso múltiple para

coordinar esas transmisiones. En este escenario, una **estación** puede ser una estación inalámbrica o un punto de acceso. Como se explica en el Capítulo 5 y en la Sección 6.2.1, en términos generales existen tres clases de protocolos de acceso múltiple: particionamiento del canal (incluyendo CDMA), acceso aleatorio y toma por turnos. Inspirados por el enorme éxito de Ethernet y su protocolo de acceso aleatorio, los diseñadores de la arquitectura 802.11 seleccionaron un protocolo de acceso aleatorio para las redes LAN inalámbricas 802.11. Este protocolo de acceso aleatorio se conoce como **CSMA con evitación de colisiones** o, más sucintamente, **CSMA/CA** (*Collision Avoidance*). Al igual que sucede con el protocolo CSMA/CD de Ethernet, las siglas "CSMA" en CSMA/CA hacen referencia al "acceso múltiple por sondeo de portadora", lo que significa que cada estación sondea el canal antes de transmitir y se abstiene de transmitir cuando detecta que el canal está ocupado. Aunque tanto Ethernet como 802.11 utilizan acceso aleatorio con sondeo de portadora, los dos protocolos MAC presentan diferencias importantes. En primer lugar, en vez de utilizar detección de colisiones, 802.11 utiliza técnicas de evitación de las colisiones. En segundo lugar, debido a las relativamente altas tasas de errores de bit de los canales inalámbricos, 802.11 (a diferencia de Ethernet) utiliza un esquema de reconocimiento/retransmisión (ARQ) de la capa de enlace. Más adelante se describen los esquemas de evitación de las colisiones y de reconocimiento en la capa de enlace de 802.11.

Recuerde de las Secciones 5.3 y 5.5 que con el algoritmo de detección de colisiones de Ethernet, una estación Ethernet escucha el canal a medida que transmite. Si detecta, mientras está transmitiendo, que hay otra estación transmitiendo también, aborta su transmisión y trata de transmitir de nuevo después de esperar un periodo de tiempo pequeño y aleatorio. A diferencia del protocolo Ethernet 802.3, el protocolo MAC 802.11 *no* implementa ningún mecanismo de detección de colisiones. Hay dos razones importantes para esto:

- La capacidad de detectar colisiones requiere la capacidad de enviar (la propia señal de la estación) y de recibir (para determinar si otra estación también está transmitiendo) al mismo tiempo. Puesto que la intensidad de la señal recibida es normalmente muy pequeña si la comparamos con la intensidad de la señal transmitida por el adaptador 802.11, resulta muy costoso construir un hardware que pueda detectar una colisión.
- Todavía más importante es que, incluso si el adaptador pudiera transmitir y escuchar al mismo tiempo (y supuestamente abortar la transmisión cuando detecte que el canal está ocupado), el adaptador seguiría sin ser capaz de detectar todas las colisiones, debido a los problemas del terminal oculto y del desvanecimiento que hemos explicado en la Sección 6.2.

Puesto que las redes LAN inalámbricas 802.11 no utilizan la detección de colisiones, una vez que una estación empieza a transmitir una trama *la transmite en su totalidad*; es decir, una vez que una estación comienza a transmitir, no hay vuelta atrás. Como cabría esperar, la transmisión de tramas completas (en especial, tramas largas) cuando las colisiones son abundantes puede degradar significativamente el rendimiento de un protocolo de acceso múltiple. Para reducir la probabilidad de colisión, 802.11 emplea varias técnicas de evitación de colisiones, que explicaremos en breve.

Sin embargo, antes de entrar en el tema de la evitación de las colisiones, necesitamos examinar en primer lugar el esquema de **reconocimiento de la capa de enlace** en 802.11. Recuerde de la Sección 6.2 que, cuando una estación en una LAN inalámbrica envía una trama, ésta puede no llegar intacta a la estación de destino por diversas razones. Para resolver esta probabilidad no desdeñable de fallo, el protocolo MAC 802.11 utiliza reconoci-

mientos de la capa de enlace. Como se muestra en la Figura 6.10, cuando la estación de destino recibe una trama, pasa la prueba de comprobación de CRC, espera un corto periodo de tiempo conocido con el nombre de **Espaciado corto entre tramas (SIFS, *Short Inter-frame Spacing*)** y luego devuelve una trama de reconocimiento. Si la estación transmisora no recibe una trama de reconocimiento dentro de un periodo de tiempo especificado, supone que se ha producido un error y retransmite la trama, utilizando el protocolo CSMA/CA para acceder al canal. Si no se recibe una trama de reconocimiento después de un número fijo de retransmisiones, la estación transmisora se da por vencida y descarta la trama.

Habiendo explorado cómo emplea 802.11 los reconocimientos de la capa de enlace, estamos en disposición de describir el protocolo CSMA/CA 802.11. Suponga que una estación (estación inalámbrica o punto de acceso) dispone de una trama para transmitir.

1. Si la estación detecta inicialmente que el canal está inactivo, transmite la trama después de un corto periodo de tiempo, conocido con el nombre de **Espacio distribuido entre tramas (DIFS, *Distributed Inter-frame Space*)**; véase la Figura 6.10.
2. En caso contrario, la estación selecciona un valor de espera (*backoff*) aleatorio y efectúa una cuenta atrás con este valor mientras detecta que el canal está inactivo. Cuando detecta que el canal está ocupado, el valor del contador permanece congelado.
3. Cuando el contador alcanza el valor cero (observe que esto sólo puede suceder mientras se detecta que el canal está inactivo), la estación transmite la trama completa y luego espera a recibir un reconocimiento.

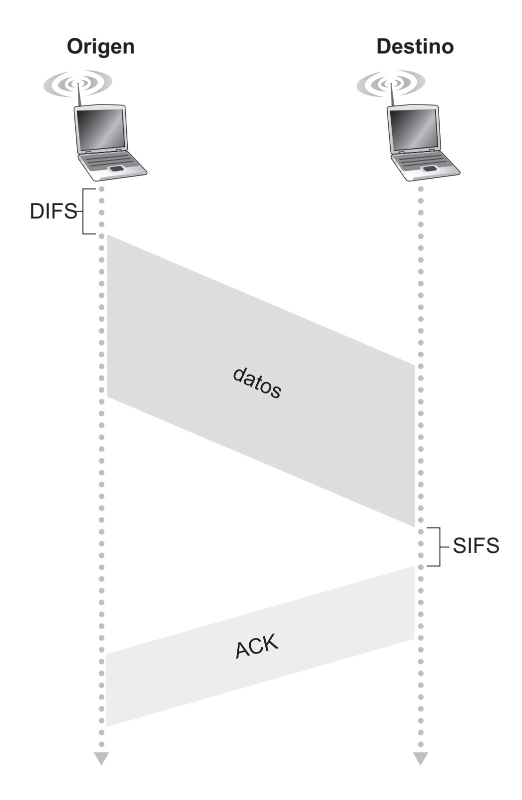


Figura 6.10 • 802.11 utiliza reconocimientos (ACK) de la capa de enlace.

4. Si se recibe una trama de reconocimiento, la estación transmisora sabe que su trama ha sido recibida correctamente en la estación de destino. Si la estación tiene otra trama que enviar, comienza de nuevo el protocolo CSMA/CA en el paso 2. Si no se recibe una trama de reconocimiento, la estación transmisora vuelva a entrar en la fase de *backoff* del paso 2, seleccionando el valor aleatorio de un intervalo más largo.

Recuerde que con el protocolo de acceso múltiple CSMA/CD de Ethernet (Sección 5.5.2) una estación comienza a transmitir en cuanto detecta que el canal está inactivo. Sin embargo, con CSMA/CA la estación se abstiene de transmitir mientras efectúa la cuenta atrás, aún cuando detecte que no hay actividad en el canal. ¿Por qué CSMA/CD y CSMA/CA adoptan enfoques tan distintos en este punto?

Para responder a esta cuestión, consideremos un escenario en el que hay dos estaciones, cada una de ellas con una trama para transmitir, pero ninguna de las estaciones transmite inmediatamente, porque ambas detectan que hay una tercera estación que ya está transmitiendo. Con CSMA/CD de Ethernet, ambas estaciones transmitirían tan pronto como detectaran que la tercera estación ha dejado de transmitir. Esto provocaría una colisión, lo cual no es un serio problema en CSMA/CD, ya que ambas estaciones abortarían sus transmisiones y por tanto evitarían transmitir inútilmente el resto de sus tramas. Sin embargo, en 802.11, la situación es muy distinta. Puesto que 802.11 no detecta las colisiones y no aborta en consecuencia las transmisiones, la trama que sufre una colisión será transmitida en su totalidad. Por tanto, el objetivo de 802.11 es evitar las colisiones siempre que sea posible. En 802.11, si las dos estaciones detectan que el canal está ocupado, ambas entran inmediatamente en un estado de espera aleatoria, con lo que cabe esperar que ambas seleccionen valores de espera (*backoff*) diferentes. Si dichos valores son verdaderamente distintos, una vez que el canal pase a estar inactivo una de las dos estaciones empezará a transmitir antes que la otra y (si las dos estaciones no están ocultas a ojos una de otra) la "estación perdedora" escuchará la señal de la estación "ganadora", congelará su cuenta atrás y se abstendrá de transmitir hasta que la estación ganadora haya completado su transmisión. De esta forma se evita una costosa colisión. Por supuesto, en 802.11 siguen pudiéndose producir colisiones con este escenario: las dos estaciones podrían estar ocultas a ojos una de otra o las dos estaciones podrían seleccionar valores de espera aleatorios lo suficientemente próximos como para que la transmisión procedente de la estación que comience primero pueda alcanzar a la segunda estación. Recuerde que ya nos hemos encontrado anteriormente con este problema al hablar de los algoritmos de acceso aleatorio en el contexto de la Figura 5.14.

Enfrentándose al problema de los terminales ocultos: RTS y CTS

El protocolo MAC 802.11 también incluye un excelente (pero opcional) esquema de reserva, que ayuda a evitar las colisiones incluso en presencia de terminales ocultos. Vamos a analizar este esquema en el contexto de la Figura 6.11, que muestra dos estaciones inalámbricas y un punto de acceso. Ambas estaciones inalámbricas caen dentro del alcance del punto de acceso (cuya área de cobertura se muestra como un círculo sombreado) y ambas están relacionadas con el punto de acceso. Sin embargo, debido al desvanecimiento, los rangos de señal de las estaciones inalámbricas están limitados al interior de los círculos sombreados de la Figura 6.11. En consecuencia, cada una de las estaciones inalámbricas está oculta a ojos de la otra, aunque ninguna de las dos está oculta para el punto de acceso.

Consideremos ahora por qué los terminales ocultos pueden ser problemáticos. Suponga que la estación H1 está transmitiendo una trama y que, en mitad de la transmisión de H1, la

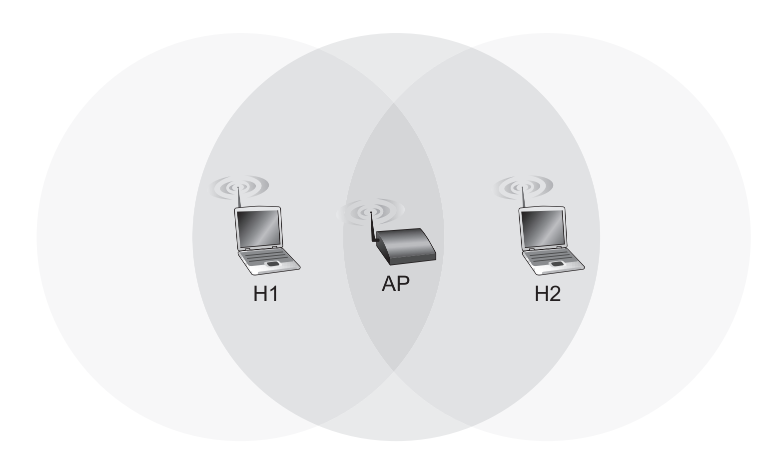


Figura 6.11 • Ejemplo de terminales ocultos: H1 está oculto a ojos de H2 y viceversa.

estación H2 quiere enviar una trama al punto de acceso. H2, al no escuchar la transmisión de H1, esperará primero un intervalo DIFS y luego transmitirá la trama, provocando una colisión. Por tanto, el canal se desperdiciará durante el periodo completo de transmisión de H1, así como durante la transmisión de H2.

Para evitar este problema, el protocolo IEEE 802.11 permite a una estación utilizar una corta trama de control de **Solicitud de transmisión (RTS, *Request to Send*)** y otra corta trama de control de **Preparado para enviar (CTS, *Clear to Send*)** para *reservar* el acceso al canal. Cuando un emisor quiere enviar una trama DATA, puede enviar primero una trama RTS al punto de acceso, indicando el tiempo total requerido para transmitir la trama DATA y la trama de reconocimiento (ACK). Cuando el punto de acceso recibe la trama RTS, responde difundiendo una trama CTS. Esta trama CTS sirve a dos propósitos distintos: proporciona al emisor un permiso explícito para enviar y también informa a las otras estaciones de que no deben transmitir durante ese periodo de tiempo reservado.

Por tanto, en la Figura 6.12, antes de transmitir una trama DATA, H1 difunde primero una trama RTS, que será escuchada por todas las estaciones situadas en su área de cobertura, incluyendo al punto de acceso. A continuación, el punto de acceso responde con una trama CTS, que será escuchada por todas las estaciones dentro de su área de cobertura, incluyendo a H1 y H2. La estación H2, habiendo escuchado la trama CTS, se abstendrá de transmitir durante el tiempo especificado en la trama CTS. Las tramas RTS, CTS, DATA y ACK se muestran en la Figura 6.12.

El uso de las tramas RTS y CTS puede mejorar el rendimiento de dos formas importantes:

- El problema de las estaciones ocultas queda mitigado, ya que una trama DATA larga sólo se transmitirá después de haber reservado el canal.
- Puesto que las tramas RTS y CTS son cortas, una colisión que implique a una trama RTS o CTS sólo durará mientras duren esas tramas cortas RTS o CTS. Una vez transmitidas las tramas RTS y CTS correctamente, las tramas DATA y ACK siguientes deberían poder transmitirse sin colisiones.

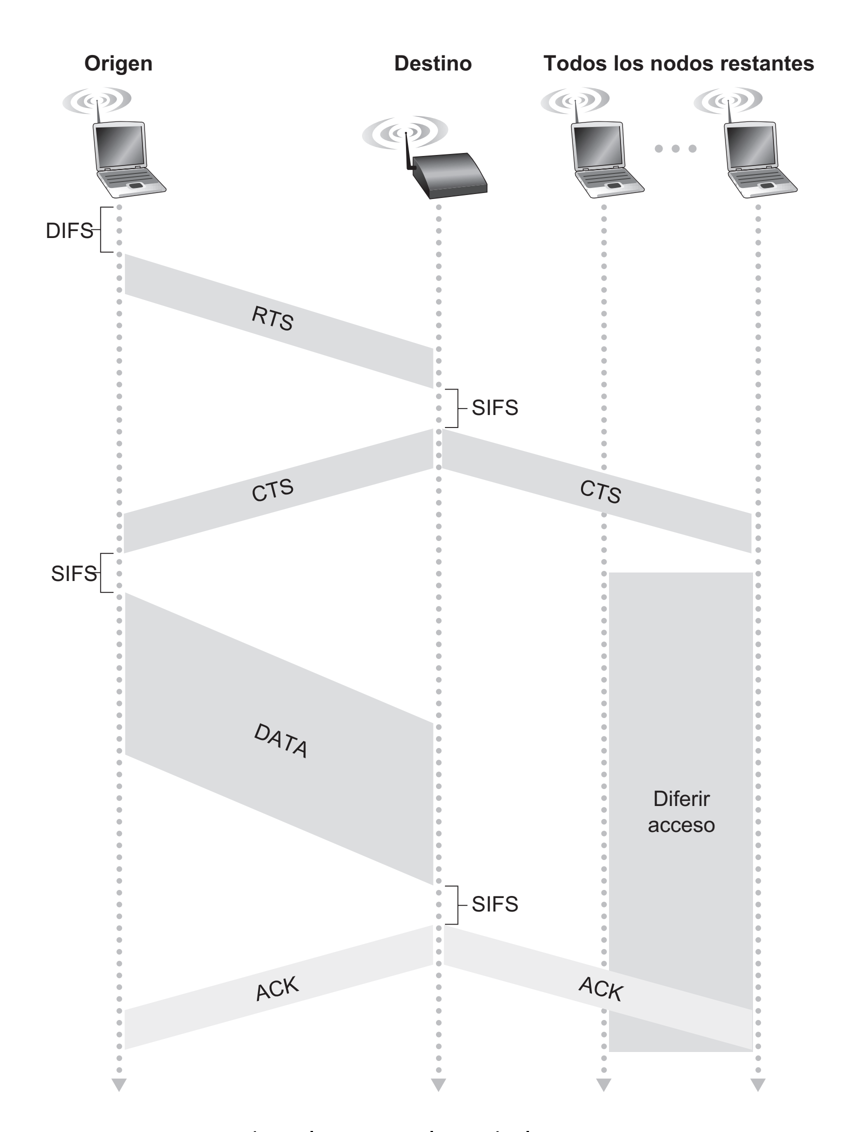


Figura 6.12 • Evitación de colisiones utilizando las tramas RTS y CTS.

Le animamos a comprobar el funcionamiento del applet 802.11 disponible en el sitio web del libro. Este applet interactivo ilustra el protocolo CSMA/CA, incluyendo la secuencia de intercambio RTS/CTS.

Aunque el intercambio RTS/CTS puede ayudar a reducir las colisiones, también introduce un retardo y consume recursos del canal. Por esta razón, el intercambio RTS/CTS solamente se utiliza (si es que se utiliza en absoluto) para reservar el canal para la transmisión de una trama DATA larga. En la práctica, cada estación inalámbrica puede establecer un umbral RTS, de modo que la secuencia RTS/CTS se utilice únicamente cuando la trama que hay que transmitir sea mayor que el umbral. Para muchas estaciones inalámbricas, el valor umbral RTS predeterminado es mayor que la longitud máxima de trama, por lo que la secuencia RTS/CTS se omite para todas las tramas DATA enviadas.

Utilización de 802.11 como un enlace punto a punto

Nuestro análisis se ha centrado hasta el momento en el uso de 802.11 en una configuración de acceso múltiple. Es necesario mencionar que si hay dos nodos, cada uno de los cuales dispone de una antena direccional, ambos pueden apuntar sus antenas hacia el otro nodo y ejecutar el protocolo 802.11 sobre lo que es, esencialmente, un enlace punto a punto. Dado el bajo coste del hardware 802.11 comercial, la utilización de antenas direccionales y una potencia de transmisión incrementada, permite utilizar 802.11 como un medio barato de proporcionar conexiones punto a punto inalámbricas a distancias de decenas de kilómetros. [Raman 2007] describe dichas redes inalámbricas multisalto operando en la llanuras rurales del Ganges en la India y que contienen enlaces 802.11 punto a punto.

6.3.3 La trama IEEE 802.11

Aunque la trama 802.11 comparte muchas similitudes con una trama Ethernet, también contiene diversos campos que son específicos para su uso en enlaces inalámbricos. La trama 802.11 se muestra en la Figura 6.13. Los números situados encima de cada campo de la trama representan las longitudes de los campos en *bytes*; los números situados por encima de cada uno de los subcampos en el campo de control de trama representan las longitudes de los subcampos en *bits*. Examinemos ahora los campos de la trama, así como algunos de los subcampos más importantes del campo de control de trama.

Campos de carga útil y CRC

En el corazón de la trama se encuentra la carga útil, que normalmente estará compuesta por un datagrama IP o un paquete ARP. Aunque el campo puede tener una longitud de hasta 2.312 bytes, normalmente la longitud es inferior a 1.500 bytes, conteniendo el campo un datagrama IP o un paquete ARP. Al igual que con una trama Ethernet, una trama 802.11 incluye un código de redundancia cíclica (CRC) de 32 bits, de modo que el receptor pueda detectar errores de bit en la trama recibida. Como ya hemos visto, los errores de bit son mucho más comunes en las redes LAN inalámbricas que en las redes LAN cableadas, por lo que el CRC aquí es todavía más útil.

Campos de dirección

Quizá la diferencia más llamativa en la trama 802.11 es que tiene *cuatro* campos de dirección, cada uno de los cuales puede contener una dirección MAC de 6 bytes. ¿Pero por qué se utilizan cuatro campos de dirección? ¿No bastaría con un campo MAC de origen y un campo MAC de destino, como sucede en Ethernet? Resulta que tres de los campos de dirección son necesarios para propósitos de la comunicación por la red, específicamente para mover el datagrama de la capa de red de una estación inalámbrica hasta una interfaz de router a través de un punto de acceso. El cuarto campo de dirección se utiliza cuando los puntos de acceso se reenvían tramas entre sí en modo ad hoc. Puesto que sólo estamos considerando aquí las redes de infraestructura, vamos a centrarnos en los tres primeros campos de dirección. El estándar 802.11 define estos campos como sigue:

- Dirección 2 es la dirección MAC de la estación que transmite la trama. Por tanto, si una estación inalámbrica transmite la trama, la dirección MAC de dicha estación se insertará en el campo Dirección 2. De forma similar, si es un punto de acceso el que trans-

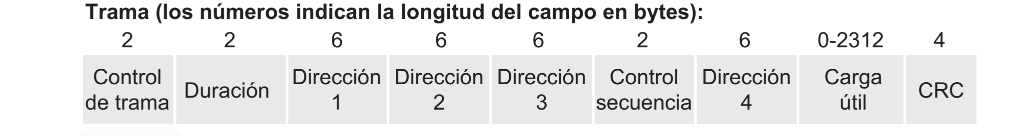


Figura 6.13 • Trama 802.11.

mite la trama, en el campo Dirección 2 se insertará la dirección MAC de dicho punto de acceso.

- El campo Dirección 1 contiene la dirección MAC de la estación inalámbrica que tiene que recibir la trama. Por tanto, si una estación inalámbrica móvil transmite la trama, Dirección 1 contendrá la dirección MAC del punto de acceso de destino. De forma similar, si un punto de acceso transmite la trama, Dirección 1 contendrá la dirección MAC de la estación inalámbrica de destino.
- Para comprender el campo Dirección 3, recuerde que el BSS (que consta del punto de acceso y las estaciones inalámbricas) forma parte de una subred y que dicha subred se conecta a otras subredes a través de alguna interfaz de router. El campo Dirección 3 contiene la dirección MAC de esa interfaz de router.

Para entender mejor el propósito del campo Dirección 3, veamos un ejemplo de comunicación por red en el contexto de la Figura 6.14. En esta figura hay dos puntos de acceso, cada uno de los cuales es responsable de una serie de estaciones inalámbricas. Cada uno de los puntos de acceso tiene una conexión directa con un router, el cual a su vez se conecta a la red Internet global. Debemos recordar que un punto de acceso es un dispositivo de la capa de enlace y que, por tanto, nunca "habla" IP ni comprende las direcciones IP. Considere ahora el proceso de transferir un datagrama desde la interfaz del router R1 hasta la estación inalámbrica H1. El router no es consciente de que existe un punto de acceso entre él y H1; desde la perspectiva del router, H1 es simplemente un host en una de las subredes a la que el router está conectado.

- El router, que conoce la dirección IP de H1 (a partir de la dirección de destino del datagrama), utiliza ARP para determinar la dirección MAC de H1, al igual que en una red LAN Ethernet normal. Después de obtener la dirección MAC de H1, la interfaz del router R1 encapsula el datagrama dentro de una trama Ethernet. El campo de dirección de origen de esta trama contiene la dirección MAC de R1 y el campo de la dirección de destino contiene la dirección MAC de H1.
- Cuando la trama Ethernet llega al punto de acceso, éste convierte la trama Ethernet 802.3 en una trama 802.11 antes de transmitirla por el canal inalámbrico. El punto de acceso rellena los campos Dirección 1 y Dirección 2 con la dirección MAC de H1 y su propia dirección MAC, respectivamente, como hemos descrito anteriormente. Como Dirección 3, el punto de acceso inserta la dirección MAC de R1. De esta manera, H1

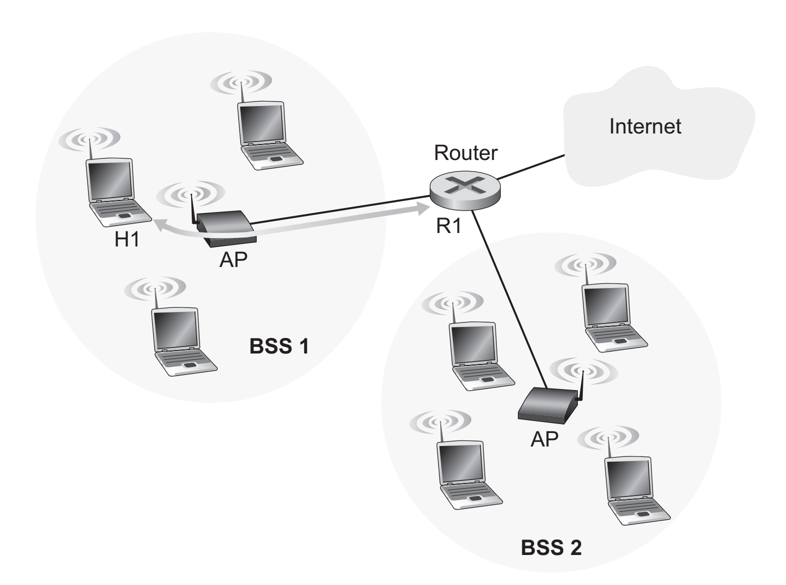


Figura 6.14 • Uso de los campos de dirección en las tramas 802.11: envío de tramas entre H1 y R1.

puede determinar (a partir de la dirección 3) la dirección MAC de la interfaz del router que envió el datagrama hacia la subred.

Considere ahora lo que sucede cuando la estación inalámbrica H1 responde transfiriendo un datagrama desde H1 a R1.

- H1 crea una trama 802.11, rellenando los campos Dirección 1 y Dirección 2 con la dirección MAC del punto de acceso y la de H1, respectivamente, como hemos descrito anteriormente. Como Dirección 3, H1 inserta la dirección MAC de R1.
- Cuando el punto de acceso recibe la trama 802.11, la convierte en una trama Ethernet. El campo de dirección de origen de esta trama será la dirección MAC de H1, mientras que el campo de dirección de destino será la dirección MAC de R1. Por tanto, el campo Dirección 3 permite al punto de acceso determinar la dirección MAC de destino apropiada a la hora de construir la trama Ethernet.

En resumen, el campo Dirección 3 desempeña un papel crucial para la comunicación por red entre el BSS y una red LAN cableada.

Campos Número de secuencia, Duración y Control de trama

Recuerde que en 802.11, cada vez que una estación recibe correctamente una trama procedente de otra estación devuelve un mensaje de reconocimiento. Puesto que los reconocimientos pueden perderse, la estación transmisora podría enviar múltiples copias de una determinada trama. Como vimos en nuestro análisis del protocolo `rdt2.1` (Sección 3.4.1), el uso de números de secuencia permite al receptor distinguir entre una trama recién transmitida y la retransmisión de una trama anterior. El campo de número de secuencia en la

trama 802.11 sirve aquí, en la capa de enlace, exactamente al mismo propósito que servía a la capa de transporte en el Capítulo 3.

Recuerde que el protocolo 802.11 permite a una estación transmisora reservar el canal durante un periodo de tiempo, que incluye el tiempo para transmitir su trama de datos y el tiempo para transmitir una trama de reconocimiento. Este valor de duración está incluido en el campo Duración de la trama (tanto para las tramas de datos como para las tramas RTS y CTS).

Como se muestra en la Figura 6.13, el campo de control de trama incluye muchos subcampos. Vamos a limitarnos a comentar algunas cosas acerca de los subcampos más importantes; para ver una exposición más completa, puede consultar la especificación 802.11 [Held 2001; Crow 1997; IEEE 802.11 1999]. Los campos *tipo* y *subtipo* se utilizan para distinguir las tramas de asociación, RTS, CTS, ACK y de datos. Los campos *hacia* y *desde* se utilizan para definir los significados de los diferentes campos de dirección (estos significados cambian dependiendo de si se está utilizando el modo ad hoc o el modo de infraestructura, y en este último caso dependiendo de si quien está enviando la trama es una estación inalámbrica o un punto de acceso). Finalmente, el campo WEP indica si se está empleando cifrado o no (hablaremos de WEP en el Capítulo 8).

6.3.4 Movilidad dentro de la misma subred IP

Para incrementar el rango físico de una red LAN inalámbrica, las empresas y universidades suelen implantar varios BSS dentro de la misma subred IP. Esto plantea naturalmente el problema de la movilidad entre los distintos BSS: ¿cómo pueden moverse las estaciones inalámbricas de forma transparente de un BSS a otro, mientras mantienen una serie de sesiones TCP activas? Como veremos en esta subsección, la movilidad puede gestionarse de forma relativamente sencilla cuando los BSS forman parte de una misma subred. Cuando las estaciones se desplazan entre subredes contiguas, hacen falta protocolos más complejos de gestión de la movilidad, como los que estudiaremos en las Secciones 6.5 y 6.6.

Examinemos ahora un ejemplo específico de movilidad entre diversos BSS dentro de la misma subred. La Figura 6.15 muestra dos BSS interconectados con un host, H1, que se desplaza desde BSS1 a BSS2. Puesto que en este ejemplo el dispositivo de interconexión que conecta a los dos BSS *no* es un router, todas las estaciones de los dos BSS, incluyendo los

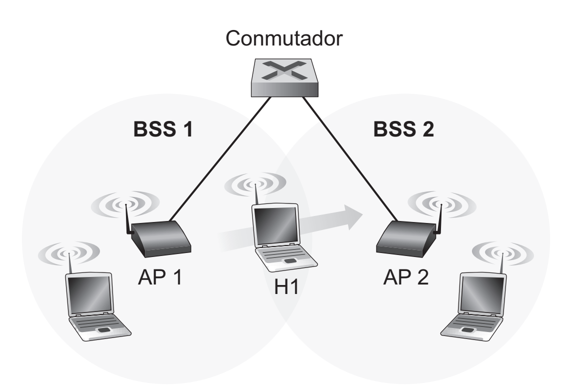


Figura 6.15 • Movilidad dentro de la misma subred.

puntos de acceso, pertenecen a la misma subred IP. Por tanto, cuando H1 se mueve desde BSS1 a BSS2 puede conservar su dirección IP y todas sus conexiones TCP activas. Si el dispositivo de interconexión fuera un router, entonces H1 tendría que obtener una nueva dirección IP en la subred hacia la cual se está moviendo. Este cambio de dirección interrumpiría (y haría que se terminara) cualquier conexión TCP activa en H1. En la Sección 6.6 veremos cómo puede utilizarse un protocolo de movilidad de la capa red, como por ejemplo IP móvil, para evitar este problema.

¿Pero qué es lo que sucede específicamente cuando H1 se mueve de BSS1 a BSS2? A medida que H1 se aleja de AP1, H1 detecta que la señal de AP1 comienza a debilitarse y empieza entonces a explorar en busca de una señal de mayor intensidad. H1 recibe tramas baliza de AP2 (que en muchos entornos corporativos y universitarios tendrán el mismo identificador SSID que AP1). H1 se desasocia entonces de AP1 y se asocia con AP2, al mismo tiempo que mantiene su dirección IP y sus sesiones TCP activas.

Esto resuelve el problema de la transferencia desde el punto de vista del host y del punto de acceso. ¿Pero qué sucede con el conmutador de la Figura 6.15? ¿Cómo sabe que el host se ha desplazado de un punto de acceso a otro? Como recordará del Capítulo 5, los conmutadores disponen de una característica de "auto-aprendizaje", que les permite construir automáticamente sus tablas de reenvío. Esta característica de auto-aprendizaje gestiona de forma eficiente los desplazamiento ocasionales (por ejemplo, cuando se transfiere a un empleado de un departamento a otro); sin embargo, los conmutadores no fueron diseñados para dar soporte a usuarios extremadamente móviles que deseen mantener las conexiones TCP mientras se desplazan entre varios BSS. Para apreciar cuál es aquí el problema, recuerde que antes del desplazamiento el conmutador tiene una entrada en su tabla de reenvío que empareja la dirección MAC de H1 con la interfaz saliente del conmutador a través de la cual se puede alcanzar a H1. Si H1 se encuentra inicialmente en BSS1, entonces un datagrama destinado a H1 tendrá que ser dirigido hacia H1 a través del punto de acceso AP1. Sin embargo, una vez que H1 se asocia con BSS2 sus tramas deben ser dirigidas hacia AP2. Una solución (que en cierto modo es un truco) es que AP2 envíe al conmutador una trama Ethernet de difusión con la dirección de origen de H1 justo después de la nueva asociación. Cuando el conmutador reciba la trama actualizará su tabla de reenvío, permitiendo alcanzar a H1 a través de AP2. El grupo de estándares 802.11f está desarrollando un protocolo entre puntos de acceso (inter-AP) para gestionar este problema y otros problemas relacionados.

6.3.5 Características avanzadas de 802.11

Vamos a terminar nuestro estudio sobre 802.11 con un breve análisis de dos capacidades avanzadas que podemos encontrar en las redes 802.11. Como veremos, estas capacidades *no* están completamente especificadas en el estándar 802.11, sino que son una serie de mecanismos especificados en el estándar los que hacen posible dichas capacidades. Esto permite que los diferentes fabricantes implementen dichas capacidades utilizando sus propias técnicas (propietarias), lo que supuestamente les puede proporcionar ventajas sobre sus competidores.

Adaptación de la velocidad en 802.11

Hemos visto anteriormente, en la Figura 6.3, que las diferentes técnicas de modulación (con las diferentes velocidades de transmisión que proporcionan) pueden resultar apropiadas para diferentes escenarios de SNR. Considere por ejemplo un usuario 802.11 móvil que

se encuentra inicialmente a 20 metros de la estación base con una alta relación señal-ruido. Dado el alto valor de SNR, el usuario puede comunicarse con la estación base utilizando una técnica de modulación de la capa física que proporcione altas velocidades de transmisión, al mismo tiempo que se mantiene una baja BER. Este usuario podrá operar en las mejores condiciones. Suponga ahora que el usuario comienza a moverse alejándose de la estación base, con lo que la SNR disminuye a medida que la distancia con la estación base se incrementa. En este caso, si no cambia la técnica de modulación utilizada en el protocolo 802.11 que opera entre la estación base y el usuario, la BER comenzará a ser inaceptablemente alta a medida que la relación SNR se reduzca, llegando eventualmente a un punto en que ninguna de las tramas transmitidas se reciba correctamente.

Por esta razón, algunas implementaciones de 802.11 tienen una capacidad de adaptación de la velocidad que permite seleccionar adaptativamente la técnica subyacente de modulación de la capa física que hay que utilizar, basándose en las características pasadas o recientes del canal. La implementación WaveLAN-II de 802.11b de Lucent [Kamerman 1997] proporciona múltiples velocidades de transmisión de datos. Si un nodo envía dos tramas consecutivas sin recibir una trama de reconocimiento (una indicación implícita de que hay errores de bit en el canal), la velocidad de transmisión se reduce al siguiente nivel inferior. Si se confirman 10 tramas consecutivas o si finaliza el recuento de un temporizador que controla el tiempo transcurrido desde la última reducción, la velocidad de transmisión se incrementa al nivel inmediatamente superior. Este mecanismo de adaptación de la velocidad comparte la misma filosofía de "prueba" que el mecanismo de control de congestión de TCP: cuando las condiciones son buenas (lo que está indicado por la recepción de tramas ACK), la velocidad de transmisión se incrementa, hasta que sucede algo "malo" (la falta de recepción de tramas ACK); cuando sucede algo "malo", la velocidad de transmisión se reduce. La adaptación de velocidad en 802.11 y en el control de congestión TCP son, por tanto, similares al niño que está constantemente pidiendo más y más a sus padres (por ejemplo, caramelos, en caso de un niño de corta edad, o una hora de llegada a casa más tardía en el caso de los adolescentes) hasta que los padres dicen finalmente que ¡Ya basta!, momento en que el niño echa marcha atrás (sólo para intentarlo posteriormente cuando la situación haya posiblemente mejorado). También se han propuesto varios otros esquemas para mejorar este sistema básico de ajuste automático de la velocidad [Holland 2001; Lacage 2004].

Gestión de la potencia

La potencia es un recurso escaso en los dispositivos móviles, por lo que el estándar 802.11 proporciona capacidades de gestión de la potencia que permiten a los nodos 802.11 minimizar la cantidad de tiempo que sus funciones de detección, transmisión y recepción, así como otros circuitos, necesitan estar "activos". La gestión de potencia en 802.11 opera de la forma siguiente: un nodo es capaz de alternar explícitamente entre los estados dormido y despierto (a diferencia de los estudiantes que se duermen durante las clases). Un nodo indica al punto de acceso que se va a ir a dormir poniendo a 1 el bit de gestión de potencia en la cabecera de una trama 802.11. Entonces se configura un temporizador en el nodo para despertar a éste justo antes del momento en el que el punto de acceso tiene programado enviar su trama baliza (recuerde que un punto de acceso envía normalmente una trama baliza cada 100 milisegundos). Puesto que el punto de acceso sabe, gracias a que el bit de gestión de potencia está activado, que el nodo se va a dormir, el punto de acceso sabrá que no debe enviar ninguna trama a dicho nodo y almacenará en un buffer todas las tramas destinadas a ese host dormido para su transmisión posterior.

El nodo se despertará justo antes de que el punto de acceso envíe una trama baliza y entrará rápidamente en el estado completamente activo (a diferencia de lo que sucede con los estudiantes adormilados, este despertar requiere únicamente 250 microsegundos [Kamerman 1997]). Las tramas baliza enviadas por el punto de acceso contienen una lista de nodos cuyas tramas se han guardado en buffer en el punto de acceso. Si no existen tramas en el buffer para el nodo, éste puede volver a dormirse. En caso contrario, el nodo puede solicitar explícitamente que se le envíen las tramas almacenadas en buffer transmitiendo un mensaje de sondeo hacia el punto de acceso. Con un tiempo entre balizas de 100 milisegundos, un tiempo de reactivación de 250 microsegundos y un tiempo similarmente pequeño para recibir una trama baliza y comprobar que no hay ninguna trama en el buffer, un nodo que no tenga tramas que enviar o recibir puede estar durmiendo el 99 por ciento del tiempo, lo que permite un ahorro considerable de energía.

6.3.6 Más allá de 802.11: Bluetooth y WiMAX

Como se ilustra en la Figura 6.2, el estándar WiFi IEEE 802.11 está pensado para la comunicación entre dispositivos separados hasta 100 metros (excepto cuando se utiliza 802.11 en una configuración punto a punto con una antena direccional). Existen otros dos protocolos estándar IEEE 802: Bluetooth (definido en el estándar IEEE 802.15.1 [IEEE 802.15 2009]) y WiMAX (definido en el estándar IEEE 802.16 [IEEE 802.16d 2004; IEEE 802.16e 2005]) para la comunicación a distancias más cortas y más largas, respectivamente.

Bluetooth

Una red IEEE 802.15.1 opera con un corto alcance, a baja potencia y con bajo coste. Se trata básicamente de una tecnología de "sustitución de cables" de baja potencia, corto alcance y baja velocidad para la interconexión de computadoras de bolsillo, dispositivos periféricos, teléfonos celulares y dispositivos PDA, mientras que 802.11 es una tecnología de "acceso" de mayor potencia, de alcance medio y mayor velocidad. Por esta razón, las redes 802.15.1 se denominan en ocasiones redes inalámbricas de área personal (WPAN, *Wireless Personal Area Network*). Las capas de enlace y física de 802.15.1 están basadas en la especificación anterior de **Bluetooth** para redes de área personal [Held 2001, Bisdikian 2001]. Las redes 802.15.1 operan en la banda de radio sin licencia de 2,4 GHz en forma TDM, con particiones de tiempo de 625 microsegundos. Durante cada partición de tiempo, un emisor transmite en uno de 79 canales, cambiando el canal de una partición a otra en una forma conocida pero pseudo-aleatoria. Este tipo de saltos de canal, que es una técnica conocida con el nombre de **Espectro disperso por salto de frecuencia (FHSS, *Frequency-Hopping Spread Spectrum*)**, distribuye las transmisiones a lo largo del tiempo por todo el espectro de frecuencias. 802.15.1 puede proporcionar velocidades de datos de hasta 4 Mbps.

Las redes 802.15.1 son redes ad hoc: no hace falta ninguna infraestructura (por ejemplo, un punto de acceso) para interconectar los dispositivos 802.15.1. Por tanto, estos dispositivos deben organizarse por sí mismos. Los dispositivos 802.15.1 se organizan primero en una **picored** (*piconet*), formada por hasta ocho dispositivos activos, como se muestra en la Figura 6.16. Uno de estos dispositivos se designa como maestro, actuando los dispositivos restantes como esclavos. El nodo maestro gobierna realmente la picored: es su reloj el que determina el tiempo en la picored; el dispositivo maestro puede transmitir en cada partición con número impar y un esclavo sólo puede transmitir después de que el maestro se haya comunicado con él en la partición anterior e incluso entonces el esclavo sólo puede transmi-

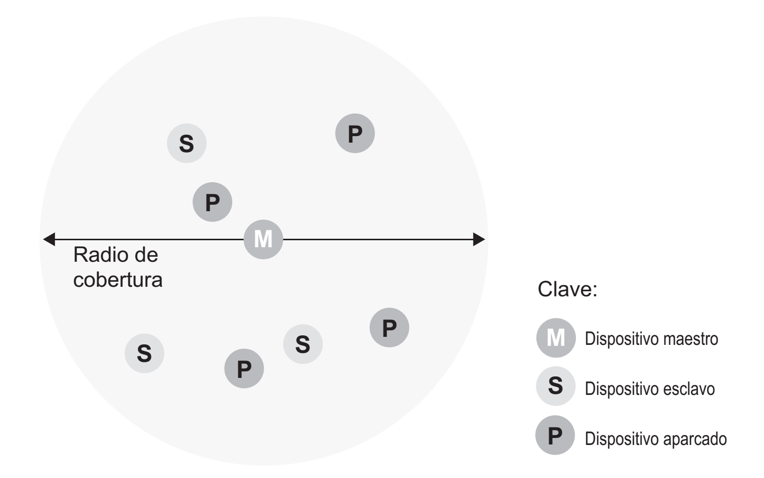


Figura 6.16 • Una picored Bluetooth.

tir hacia el maestro. Además de los dispositivos esclavos, puede haber hasta 255 dispositivos aparcados dentro de la red. Estos dispositivos no pueden comunicarse hasta que su estado sea cambiado por el nodo maestro de aparcado a activo.

Para obtener más información acerca de las redes WPAN 802.15.1, el lector interesado puede consultar las referencias sobre Bluetooth [Held 2001, Bisdikian 2001] o el sitio web oficial de IEEE 802.15 [IEEE 802.15 2009].

WiMAX

WiMAX (*World Interoperability for Microwave Access,* Interoperabilidad mundial para acceso por microondas) es una familia de estándares IEEE 802.16 que trata de suministrar datos inalámbricos a un gran número de usuarios en un área extensa, a velocidades capaces de competir con las de los modems por cable y las redes ADSL. El estándar 802.16d actualiza el estándar anterior 802.16a. El estándar 802.16e pretende soportar la movilidad a velocidades de 70-80 millas por hora, unos 105-120 kilómetros por hora (es decir, velocidad de autopista en la mayoría de los países fuera de Europa) y tiene una estructura de enlace diferente para dispositivos pequeños y de recursos limitados tales como las PDA, los teléfonos y las computadoras portátiles.

La arquitectura 802.16 está basada en la noción de una estación base que sirve de modo centralizado a un número potencialmente grande de clientes (conocidos con el nombre de estaciones de abonado) asociados con dicha estación base. En este sentido, WiMAX recuerda tanto a WiFi en modo de infraestructura como a las redes de telefonía celular. La estación base coordina la transmisión de los paquetes de la capa de enlace tanto en la dirección de bajada (desde la estación base a las estaciones de abonado) como de subida (desde las estaciones de abonado hacia la estación base), de acuerdo con la estructura de marcos TDM mostrada en la Figura 6.17. Utilizaremos aquí el término “paquete” en lugar del término “trama” (que hemos usado para 802.11 y otros paquetes de la capa de enlace) para distinguir la unidad de datos de la capa de enlace de la estructura de marcos TDM mostrada en la Figura 6.17. WiMAX opera, por tanto, en modo de multiplexación por división en el tiempo (TDM), aunque los tiempos de marco son variables como se indica más adelante.

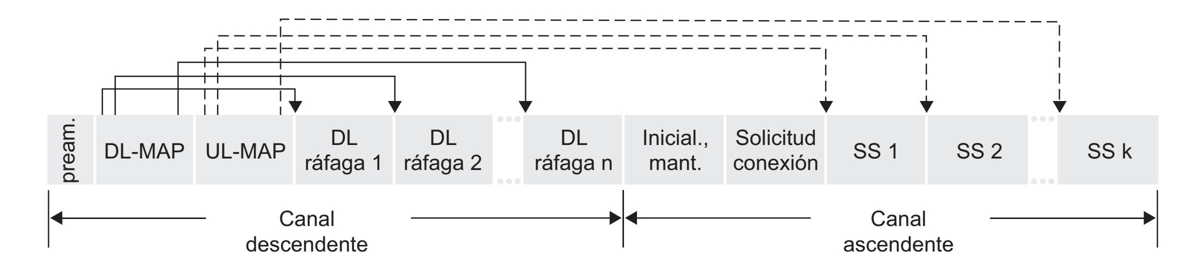


Figura 6.17 • Estructura de marcos TDM 802.16.

Observemos que WiMAX también define un modo FDM de operación, aunque aquí no lo vamos a tratar.

Al principio de la estructura de marcos TDM, la estación base envía en primer lugar la lista de mensajes MAP (*Media Access Protocol*) descendentes que informa a las estaciones de abonado de las propiedades de la capa física (esquema de modulación, codificación y parámetros de corrección de errores) que se utilizarán para transmitir las subsiguientes ráfagas de paquetes dentro de la estructura de marcos TDM. Puede haber múltiples ráfagas dentro de una estructura y múltiples paquetes dentro de una ráfaga destinada a una determinada estación de abonado. La estación base transmite todos los paquetes contenidos en la ráfaga utilizando las mismas propiedades de la capa física. Sin embargo, dichas propiedades pueden cambiar de una ráfaga a otra, permitiendo a la estación base seleccionar esquemas de transmisión de la capa física que estén óptimamente adaptados a cada estación de abonado receptora. La estación base puede seleccionar el conjunto de receptores a los que va a enviar en esta estructura en función de las condiciones actuales estimadas del canal existente hasta cada receptor. Esta forma de **planificación oportunista** [Bender 2000, Kulkarni 2005] (adaptar el protocolo de la capa física a las condiciones del canal entre el emisor y el receptor, y seleccionar los receptores a los que se enviarán los paquetes basándose en las condiciones del canal) permite a la estación base hacer un uso óptimo del medio inalámbrico. El estándar WiMAX no impone un conjunto concreto de parámetros de la capa física que haya que utilizar en una situación determinada; dicha decisión se deja al arbitrio del fabricante del equipo WiMAX y del operador de red.

Una estación base WiMAX también regula el acceso de las estaciones de abonado al canal ascendente mediante el uso de mensajes UL-MAP. Estos mensajes controlan la cantidad de tiempo durante la cual se da a cada estación de abonado acceso al canal en los subsiguientes canales ascendentes. De nuevo, el estándar WiMAX no impone ninguna política concreta para asignar el tiempo de canal ascendente a un cliente; es una decisión que se deja al operador de red. En lugar de ello, WiMAX proporciona los mecanismos (tales como los mensajes de control UL-MAP) para implementar una política que podría asignar diferentes periodos de tiempo de acceso al canal a las distintas estaciones de abonado. Los abonados emplean las partes iniciales de la estructura de canal ascendente para transmitir mensajes de control del enlace radio, mensajes para solicitar la admisión y la autenticación en la red WiMAX y mensajes de protocolos de mayor nivel relacionados con la administración, como por ejemplo DHCP y SNMP.

La Figura 6.18 muestra el formato del paquete MAC WiMAX. El único campo que vamos a comentar aquí es el campo de identificador de conexión de la cabecera. WiMAX es una arquitectura orientada a conexión que permite que cada conexión tenga asociados una calidad de servicio (QoS), unos parámetros de tráfico y otras informaciones. El cómo haya

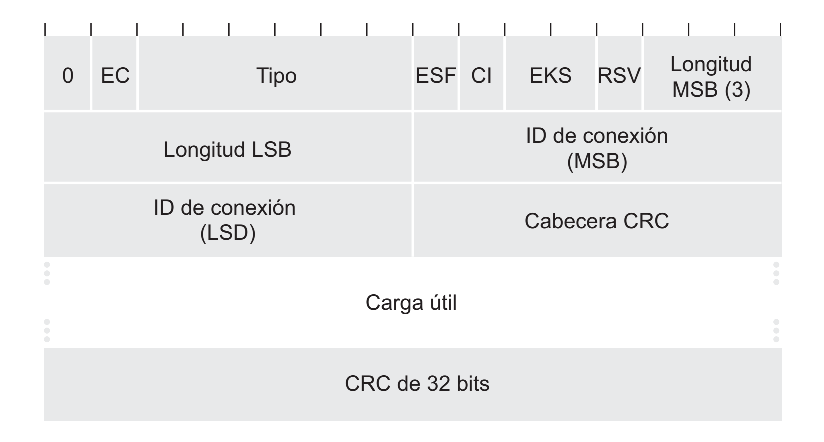


Figura 6.18 • Paquete 802.16.

que proporcionar esta calidad de servicio es cuestión del operador de red. WiMAX proporciona los mecanismos de bajo nivel (por ejemplo, estimación del canal y campos de solicitud de admisión para la conexión, con el fin de transportar información entre la estación base y el host), pero no proporciona el enfoque global ni las políticas para proporcionar la calidad de servicio. Aún cuando cada estación de abonado tendrá una dirección MAC de 48 bits (como en las redes 802.3 y 802.11), en WiMAX esta dirección MAC se puede considerar más apropiadamente como un identificador de equipo, ya que la comunicación entre equipos terminales termina por ser asignada a un identificador de conexión (en lugar de a las direcciones de los terminales emisor y receptor de la conexión).

Nuestro tratamiento de WiMAX ha sido necesariamente breve y hay muchos otros temas que no hemos podido analizar aquí, como la gestión de potencia (un modo de dormir similar al de 802.11), la transferencia de llamadas (*handoff*), la planificación dependiente del estado del canal de las transmisiones de unidades de datos protocolo (PDU) MAC procedentes de la estación base, el mecanismo QoS y la seguridad. Estando el estándar 802.16e todavía en desarrollo, los sistemas WiMAX continuarán evolucionando a lo largo de los próximos años. Mientras que los estándares indicados anteriormente permiten conocer de una forma relativamente "ardua" éstos y otros temas relacionados con WiMAX, [Eklund 2002, Cicconetti 2006] proporcionan una serie de panorámicas de WiMAX muy comprensibles.

6.4 Acceso celular a Internet

En la sección anterior hemos examinado cómo un host puede acceder a Internet cuando entra dentro de un área de cobertura WiFi, es decir, cuando se encuentra en las vecindades de un punto de acceso 802.11. Pero la mayoría de las áreas WiFi tienen una cobertura pequeña, de entre 10 y 100 metros de diámetro y las redes WiMAX, que tienen un área mayor, están todavía por implantarse. ¿Qué podemos hacer cuando necesitamos desesperadamente acceso inalámbrico a Internet y no podemos acceder a un área WiFi?

Dado que la telefonía celular es ahora onmipresente en muchas áreas de todo el mundo, una estrategia natural consiste en extender las redes celulares de modo que soporten no sólo la telefonía de voz sino también el acceso inalámbrico a Internet. Idealmente, este acceso a

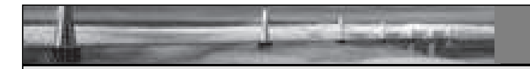
Internet se llevaría a cabo a una velocidad razonablemente alta y permitiría una movilidad transparente, con lo que los usuarios podrían mantener sus sesiones TCP mientras están viajando, por ejemplo, en un autobús o en un tren. Con tasas de bit lo suficientemente altas tanto para subida como para bajada, el usuario podría incluso mantener sesiones de videoconferencia mientras deambula de un lado a otro (*roaming*). Este escenario no es tan futurista como puede parecer. En el momento de escribir estas líneas (primavera de 2009), muchos proveedores de servicios de telefonía celular de Estados Unidos ofrecen a sus abonados un servicio de acceso celular a Internet por menos de 50 dólares mensuales, con velocidades de bajada y de subida de unos cuantos centenares de kilobits por segundo. Asimismo, están empezando a estar disponibles velocidades de datos de varios megabits por segundo a medida que se van implantando cada vez más los servicios de datos de banda ancha como HSDPA.

En esta sección, proporcionamos una breve panorámica de las tecnologías actuales y emergentes de acceso celular a Internet. Nos centraremos de nuevo principalmente en el primer salto inalámbrico entre el teléfono celular y la infraestructura de red cableada; en la Sección 6.7 veremos cómo se enrutan las llamadas hacia un usuario que se está moviendo entre las distintas estaciones base. Nuestro análisis, debido a su brevedad, sólo proporcionará una descripción simplificada y de nivel general de las tecnologías celulares. Por supuesto, las comunicaciones celulares modernas son un tema de gran profundidad y de gran amplitud, existiendo muchas universidades que ofrecen cursos completos sobre el mismo. Los lectores que deseen conocer más detalles pueden consultar [Goodman 1997; Kaaranen 2001; Lin 2001; Korhonen 2003, Schiller 2003; Scourias 2007; Turner 2009], así como [Mouly 1992] que es una referencia particularmente excelente y exhaustiva.

6.4.1 Panorámica de la arquitectura de las redes celulares

En nuestra descripción de la arquitectura de las redes celulares de esta sección, adoptaremos la terminología de los estándares del *Sistema global de comunicaciones móviles (GSM, Global System for Mobile Communications)*. (Para los aficionados a las cuestiones históricas, el acrónimo GSM derivaba originalmente de *Groupe Spécial Mobile*, aunque luego se adoptó el nombre inglés, conservándose las letras originales del acrónimo.) En la década de 1980, los organismos reguladores europeos se dieron cuenta de la necesidad de un sistema de telefonía celular digital europeo que pudiera sustituir los numerosos sistemas de telefonía celular analógica, que eran incompatibles entre sí. Esta iniciativa condujo a la definición del estándar GSM [Mouly 1992]. En Europa se implantó la tecnología GSM con un gran éxito a principios de la década de 1990, y desde entonces GSM ha crecido hasta convertirse en el sistema dominante dentro de la telefonía celular; en estos momentos, más del 80 por ciento de todos los abonados de telefonía celular del mundo utilizan GSM.

Cuando las personas hablan acerca de la tecnología celular, a menudo la clasifican como perteneciente a una de varias "generaciones". Las primeras generaciones estaban diseñadas principalmente para el tráfico de voz. Los sistemas de primera generación (1G) eran sistemas FDMA analógicos, diseñados exclusivamente para la comunicación únicamente de voz. Estos sistemas 1G están prácticamente extintos en la actualidad, habiendo sido sustituidos por los sistemas 2G digitales. Los sistemas 2G originales también estaban diseñados para voz, pero posteriormente se ampliaron (2.5G) para soportar servicios tanto de voz como de datos (es decir, Internet). Los sistemas 3G, que están siendo implantados actualmente, también soportan voz y datos, pero poniendo un énfasis cada vez mayor en las capacidades de datos y en los enlaces de acceso de radio de más alta velocidad.



HISTORIA

TECNOLOGÍA MÓVIL CELULAR 3G FRENTE A REDES LAN INALÁMBRICAS

Muchos operadores de telefonía móvil celular están implantando sistemas móviles celulares 3G con velocidades de datos en interiores de 2 Mbps y en exteriores de 384 kbps y superiores. Los sistemas 3G se están implantando en bandas de radiofrecuencia con licencia, habiendo pagado algunos operadores sumas considerables a los respectivos gobiernos por dichas licencias. Los sistemas 3G permiten a los usuarios acceder a Internet desde ubicaciones remotas en exteriores mientras están viajando, de forma similar al actual acceso a los servicios de telefonía celular. Por ejemplo, la tecnología 3G permite a un usuario acceder a información de mapas de carreteras mientras está conduciendo un vehículo o consultar la información sobre la cartelera cinematográfica mientras está tomando el sol en una playa. Sin embargo, muchos expertos están empezando a cuestionar si la tecnología 3G tendrá éxito, dado su alto coste y la competencia representada por la tecnología de redes LAN inalámbricas. En particular, estos expertos argumentan que:

- La emergente infraestructura de redes LAN inalámbricas será en el futuro prácticamente ubicua. Las redes LAN inalámbricas IEEE 802.11, que operan a 54 Mbps, disfrutan de una implantación cada vez mayor. Casi todas las computadoras portátiles y las PDA vienen equipadas de fábrica con tarjetas LAN 802.11. Además, los dispositivos Internet emergentes (como las cámaras inalámbricas y los marcos de fotografías electrónicos) también utilizarán pequeñas tarjetas LAN inalámbricas de baja potencia.
- WiMAX, que hemos estudiado en la Sección 6.3.6, promete ofrecer servicios de datos de área extensa a los usuarios móviles a velocidades de varios megabits por segundo o superiores. Sprint Nextel está invirtiendo muchos miles de millones de dólares en la implantación de WiMAX.
- Las estaciones base para redes LAN inalámbricas podrían también comunicarse con dispositivos de telefonía móvil. Los teléfonos futuros podrían ser capaces de conectarse a la red de telefonía celular o a una red IP, utilizando un servicio de voz sobre IP similar a Skype, evitando con ello los servicios de datos 3G y los servicios de voz de telefonía celular de los operadores.

Por supuesto, muchos otros expertos piensan que 3G no sólo será un gran éxito, sino que tambien va a revolucionar de forma importante la manera en que trabajamos y vivimos. Evidentemente, puede que tanto WiFi como 3G se conviertan en tecnologías inalámbricas prevalentes, y que los dispositivos inalámbricos puedan seleccionar automáticamente, mientras se desplazan, la tecnología de acceso que proporcione el mejor servicio en cada ubicación física concreta.

Arquitectura de redes celulares, 2G: conexiones de voz con la red telefónica

El término *celular* hace referencia al hecho de que la región cubierta por una red celular está dividida en una serie de áreas geográficas de cobertura, denominadas **celdas**, que se muestran como hexágonos en la parte izquierda de la Figura 6.19. Igual que el estándar WiFi 802.11 que hemos estudiado en la Sección 6.3.1, GSM tiene su propia nomenclatura parti-

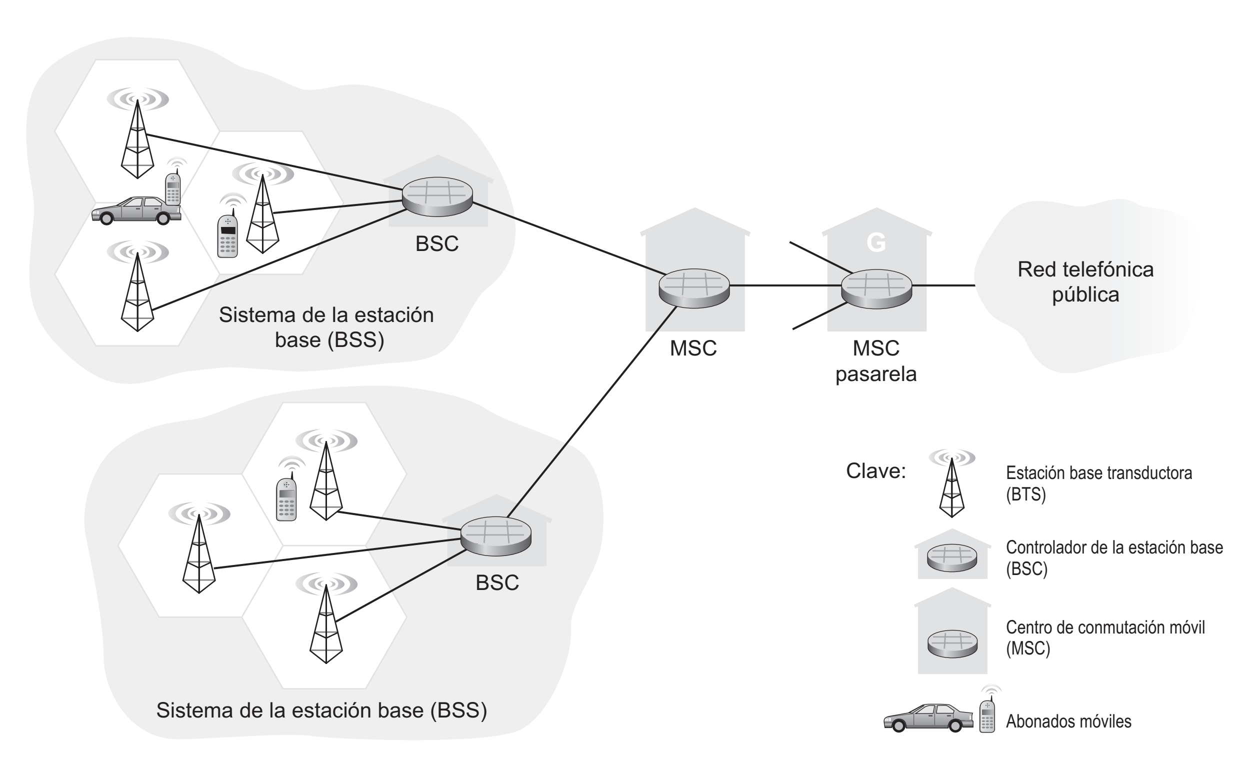


Figura 6.19 • Componentes de una arquitectura de red celular GSM 2G.

cular. Cada celda contiene una **estación transductora base (BTS, *Base Transceiver Station*)** que transmite y recibe señales hacia y desde las estaciones móviles que se encuentran dentro de su celda. El área de cobertura de una celda depende de muchos factores, incluyendo la potencia de transmisión de la BTS, la potencia de transmisión de los dispositivos de usuario, los edificios situados dentro de la celda que puedan obstruir las comunicaciones y la altura de las antenas de la estación base. Aunque la Figura 6.19 muestra que cada celda contiene una estación transductora base situada en el centro de la celda, muchos sistemas actuales colocan las BTS en las esquinas donde intersectan tres celdas, de modo que una única BTS con antenas direccionales pueda dar servicio a las tres.

El estándar GSM para los sistemas celulares 2G utiliza una combinación FDM/TDM (radio) para la interfaz aérea. Recuerde del Capítulo 1 que, con FDM pura, el canal se particiona en una serie de canales de frecuencia, estando cada banda dedicada a una llamada. Recuerde también de ese capítulo que, con la multiplexación TDM pura, el tiempo se divide en marcos que a su vez se subdividen en particiones y que a cada llamada se le asigna el uso de una partición concreta dentro del marco. En los sistemas FDM/TDM combinados, el canal se particiona en una serie de sub-bandas de frecuencia y dentro de cada sub-banda el tiempo se divide en marcos y particiones. Por tanto, para un sistema FDM/TDM combinado, si el canal está particionado en F sub-bandas y el tiempo se divide en T particiones, entonces el canal podrá soportar $F \cdot T$ llamadas simultáneas.

Los sistemas GSM están compuestos por bandas de frecuencia de 200 kHz soportando cada banda ocho llamadas TDM. GSM codifica la voz a 13 kbps y a 12,2 kbps. Un estándar competidor de GSM, IS-95 CDMA y su sucesor CDMA 2000, utiliza acceso múltiple

por división de código (véase la Sección 6.2.1) en lugar de la técnica combinada FDM/TDM, lo que hace que los teléfonos de usuario GSM no puedan operar en una red IS-95 y *viceversa.*

El **controlador de la estación base (BSC, *Base Station Controller*)** de una red GSM puede estar físicamente ubicado junto a una BTS, pero normalmente un mismo BSC dará servicio a varias decenas de estaciones transductoras base. El papel del BSC consiste en asignar los canales de radio de las BTS a los abonados móviles, realizar las tareas de **localización de abonados (*paging*)** (determinar la celda en la que se encuentra un usuario móvil) y llevar a cabo la transferencia de los usuarios móviles, que es un tema del que nos ocuparemos en breve en la Sección 6.7.2. El controlador de las estaciones base y las estaciones transductoras base que controla forman, colectivamente, un **Sistema de estaciones base (BSS, *Base Station System*)** GSM.

Como veremos en la Sección 6.7, el **Centro de conmutación móvil (MSC, *Mobile Switching Center*)** desempeña el papel central en lo que respecta a la autorización de los usuarios y la facturación (por ejemplo, determinando si se permite a un cierto dispositivo móvil conectarse a la red celular), el establecimiento y finalización de llamadas y la transferencia de las mismas. Un único MSC contendrá normalmente hasta cinco BSC, lo que permite tener aproximadamente unos 200.000 abonados por MSC. La red de un proveedor de servicio celular tendrá un serie de centros MSC, siendo algunos de ellos MSC especiales conocidos con el nombre de centros MSC pasarela, los cuales sirven para conectar la red celular de ese proveedor con la red telefónica pública de mayor tamaño.

Arquitectura de las redes celulares, 2.5G y 3G: ampliación de Internet a los abonados de telefonía celular

Hasta el momento, nuestras explicaciones se han centrado en la conexión de los usuarios de telefonía celular a la red de telefonía pública. Pero, cada vez más, los usuarios móviles acceden a Internet a través de la red celular utilizando dispositivos como iPhones, Blackberries, computadoras portátiles y otros. Una forma de hacer esto utilizando sólo la infraestructura 2G mostrada en la Figura 6.19 consiste en utilizar una conexión de telefonía celular como conexión de acceso telefónico a un ISP, de la misma forma que muchos usuarios domésticos utilizaban en la década de 1990 su teléfono convencional para tener acceso telefónico a un ISP. Sin embargo, la desventaja de este enfoque es la velocidad de transmisión extremadamete lenta (normalmente unas decenas de kilobits por segundo y a menudo menos que eso) disponible mediante una conexión celular de acceso telefónico. Idealmente, lo que nos gustaría es ampliar el alcance de IP hasta el propio sistema de estaciones base utilizando líneas de alto ancho de banda y emplear después múltiples canales de voz o redes mejoradas de acceso por radio para conectar a los usuarios móviles al sistema de estaciones base a alta velocidad. Éste es precisamente el enfoque adoptado en los sistemas celulares 2.5G y 3G.

La Figura 6.20 muestra la arquitectura de red celular para 2.5G GSM, que amplía el estándar 2G GSM para proporcionar acceso de alta velocidad a Internet. El enfoque adoptado por los diseñadores de 2.5G GSM es muy claro: *no tocar el núcleo de la red de telefonía celular GSM*. Esto se consigue proporcionando acceso a Internet en la frontera de la red, como una funcionalidad añadida independiente, en lugar de como una funcionalidad integrada en el núcleo de la red de telefonía celular existente (lo que hubiera requerido efectuar cambios en dicho núcleo). Esa capacidad añadida se implementa en la red de acceso por radio, el BSC, mediante la introducción de una red separada de nodos **SGSN (*Serving GPRS Support Node*, Nodo de soporte GPRS)**.

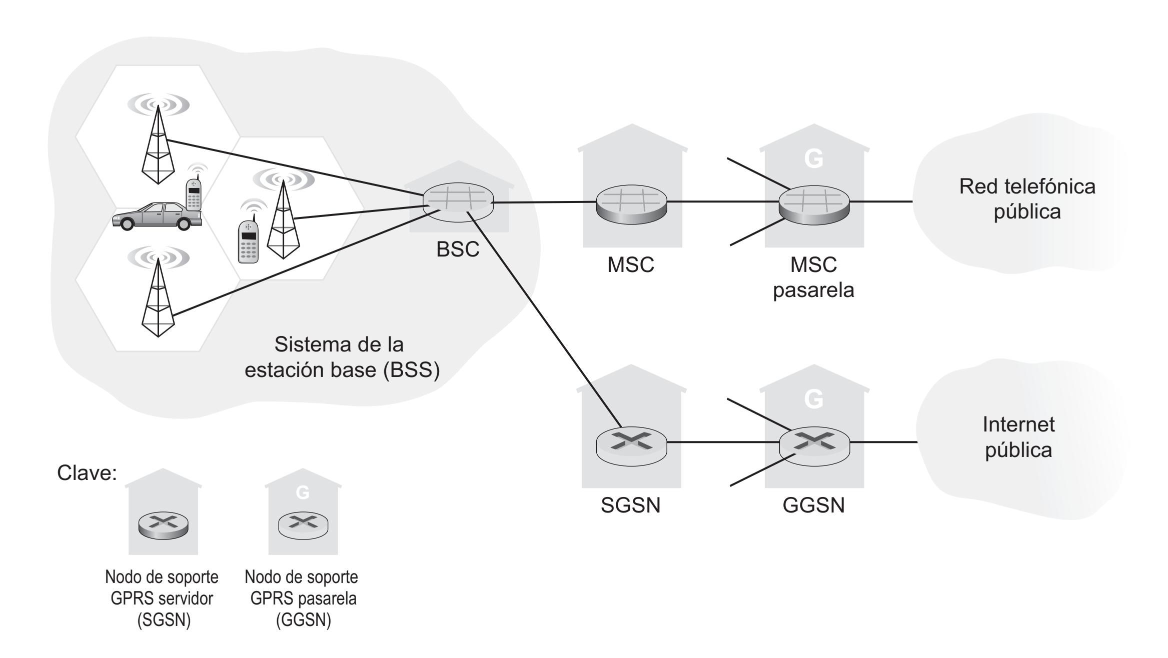


Figura 6.20 • Arquitectura de las redes de voz y datos celulares GSM 2.5G.

En el BSC, los canales FDM/TDM de la interfaz aérea que transportan datagramas IP se reenvían desde el BSC hacia el nodo SGSN el cual se comunica con el MSC para llevar a cabo la autorización de los usuarios, la transferencia de los mismos y otras funciones. Además de este tráfico de señalización, el SGSN se encarga de reenviar los datagramas IP del BSC hacia/desde Internet. En la red de acceso por radio, se introdujo el **Servicio general de paquetes radio (GPRS, *General Packet Radio Service*)** en GSM 2.5G para permitir a los usuarios utilizar dinámicamente múltiples canales de radio para los datos IP, obteniéndose así tasas de hasta 115 kbps. Siguiendo la estela de GPRS, luego se introdujo el sistema ***Velocidades de datos mejoradas para evolución global (EDGE, Enhanced Data Rates for Global Evolution)*** con el fin de incrementar las velocidades de datos de una red GSM/GPRS hasta 384 kbps. [Ericsson 2009] proporciona una excelente panorámica de EDGE.

Los sistemas celulares 3G ofrecen servicios de telefonía y velocidades de datos significativamente mayores que sus contrapartidas 2.5G. En particular, los sistemas 3G están obligados a proporcionar:

- 144 kbps a velocidades de conducción de vehículos.
- 384 kbps para utilización estática en exteriores o velocidades típicas de una persona caminando.
- 2 Mbps para interiores.

El **Servicio universal de comunicaciones móviles (UMTS, *Universal Mobile Telecommunications Service*)**, una de las tecnologías 3G más populares, es una evolución de GSM 2.5G que soporta capacidades 3G. La arquitectura de una red UMTS se asemeja bastante a la arquitectura de las redes GSM establecidas. En particular, se continúan utilizando las redes de datos 2.5G existentes que se muestran en la Figura 6.20, de la misma manera

que las redes 2.5G respetaban las redes de voz existentes en el momento de su introducción. Un cambio significativo en UMTS es que, en lugar de utilizar el esquema FDMA/TDMA de GSM, utiliza una técnica CDMA, denominada CDMA de banda ancha mediante secuencia directa (DS-WCDMA, *Direct Sequence Wideband* CDMA), dentro de particiones TDMA (con marcos de particiones TDMA disponibles en múltiples frecuencias, lo que constituye un uso interesante de las tres técnicas dedicadas de compartición de un canal que hemos identificado anteriormente). Este cambio requiere una nueva red de acceso inalámbrico celular operando en paralelo con la red de sistemas BSS mostrada en la Figura 6.20. El servicio de datos asociado con la especificación WCDMA se conoce con las siglas HSDPA/HSUPA (*High Speed Downlink/Uplink Packet Access*, Acceso de paquetes de alta velocidad descendente/ascendente) y promete velocidades de datos de hasta 14 Mbps. Puede encontrar más detalles relativos a las redes 3G en el sitio web del Proyecto del Consorcio de Tercera Generación (3GPP, *3rd Generation Partnership Project*) [3GPP 2009].

En junio de 2007 ya habían sido conectados 200 millones de abonados 3G. Esto sólo representa el 6,7 por ciento de los 3.000 millones de abonados de telefonía móvil que hay en todo el mundo. En los países en los que los sistemas 3G fueron introducidos primero (Japón y Corea del Sur) más de la mitad de todos los abonados utilizan 3G. En Europa, el país puntero es Italia, con un tercio de sus abonados utilizando 3G. Otros países punteros incluyen Reino Unido, Austria, Australia y Singapur.

Habiéndose definido varias generaciones de especificaciones 3G y estando en marcha las correspondientes implementaciones, ¿tendremos que esperar mucho para que aparezcan los sistemas inalámbricos 4G? La respuesta es sí y no. No existe ninguna definición formal de cómo deben ser esos sistemas 4G, a pesar de lo cual hay fabricantes que están diseñando equipos (por ejemplo, WiMAX) que ya superan las prestaciones de los sistemas 3G. Por supuesto, cuando se definan e implementen los sistemas 4G, estos operarán a velocidades más altas que los sistemas 3G, de 1 Gbps o más; asimismo, permitirán integrar más estrechamente los protocolos de Internet y lo más probable es que se centren en las comunicaciones multimedia, los servicios basados en la ubicación y en la seguridad.

6.5 Gestión de la movilidad: principios

Habiendo cubierto la naturaleza *inalámbrica* de los enlaces de comunicaciones existentes en una red inalámbrica, ahora es el momento de volver nuestra atención hacia la *movilidad* que estos enlaces inalámbricos permiten. En el sentido más amplio, un nodo móvil es aquél que cambia su punto de conexión con la red a lo largo del tiempo. Puesto que el término *movilidad* ha adoptado muchos significados tanto en el mundo de las computadoras como en el de la telefonía, conviene primero considerar con cierto detalle diversas dimensiones de la movilidad.

- *Desde el punto de vista de la capa de red, ¿cómo de móvil es un usuario?* Un usuario físicamente móvil planteará un conjunto muy distinto de desafíos a la capa de red, dependiendo de cómo se mueva entre los puntos de conexión con la red. En un extremo del espectro de la Figura 6.21, un usuario puede llevar consigo una computadora portátil con una tarjeta de interfaz de red inalámbrica mientras pasea por un edificio. Como vimos en la Sección 6.3.4, este usuario *no* es móvil desde la perspectiva de la capa de red. Además, si el usuario se asocia con el mismo punto de acceso, independientemente

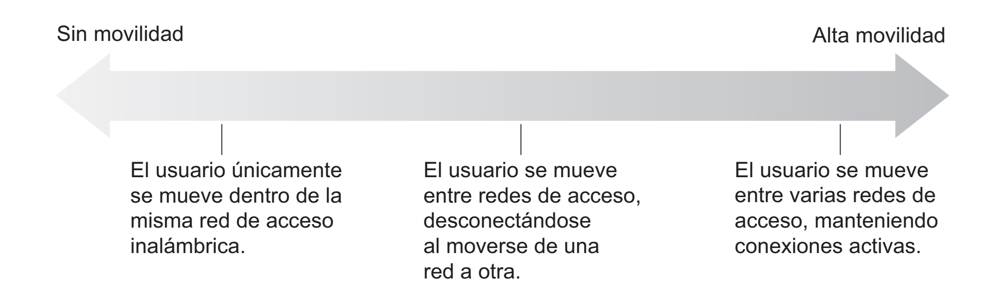


Figura 6.21 • Diversos grados de movilidad desde el punto de vista de la capa de red.

de su ubicación, ese usuario no es ni siquiera móvil desde la perspectiva de la capa de enlace.

En el otro extremo del espectro, considere el usuario que está viajando a 150 kilómetros por hora por una autovía en un BMW, pasando por múltiples redes de acceso inalámbricas y que quiere mantener una conexión TCP ininterrumpida con alguna aplicación remota a todo lo largo del viaje. ¡Este usuario sí que es móvil sin ningún tipo de dudas! Entre estos dos extremos se encontraría el usuario que traslada una computadora portátil desde una ubicación (por ejemplo, su oficina o su domicilio) a otra (por ejemplo, una cafetería o un aula) y quiere conectarse a la red en esa nueva ubicación. Este usuario también es móvil (aunque menos que el conductor del BMW), pero no necesita mantener una conexión activa mientras se está moviendo entre los puntos de conexión con la red. La Figura 6.21 ilustra este espectro de movilidad del usuario desde la perspectiva de la capa de red.

- *¿Hasta qué punto es importante que la dirección del nodo móvil sea siempre la misma?* Con la telefonía móvil, nuestro número de teléfono (que es básicamente la dirección de la capa de red de nuestro teléfono) continúa siendo el mismo a medida que nos desplazamos desde la red de telefonía móvil de un proveedor a la de otro. ¿Deben las computadoras portátiles mantener de forma similar la misma dirección IP mientras se están desplazando entre redes IP?

  La respuesta a esta cuestión dependerá en gran medida de las aplicaciones que se estén ejecutando. Para el conductor del BMW que quiere mantener una conexión TCP ininterrumpida con una aplicación remota mientras viaja por la autovía, sería conveniente mantener la misma dirección IP. Recuerde del Capítulo 3 que una aplicación Internet necesita conocer la dirección IP y el número de puerto de la entidad remota con la que se está comunicando. Si una entidad móvil es capaz de mantener su dirección IP a medida que se desplaza, la movilidad se convertirá en algo transparente desde el punto de vista de la aplicación. Esta transparencia tiene un enorme valor, ya que la aplicación no tendrá que preocuparse de la cuestión de que las direcciones IP puedan potencialmente cambiar, y un mismo código de aplicación permitirá dar servicio tanto a conexiones móviles como no móviles. Veremos en la siguiente sección que la tecnología de IP móvil proporciona esta transparencia, permitiendo a un nodo móvil mantener su dirección IP permanente mientras se está desplazando de una red a otra.

  Por otro lado, un usuario móvil menos glamuroso podría simplemente querer apagar su computadora portátil en la oficina, llevarla a su domicilio, volver a encenderla y trabajar desde su casa. Si la computadora portátil funciona principalmente como un cliente en

aplicaciones cliente-servidor (por ejemplo, enviar/leer correo electrónico, navegar por la Web, conectarse mediante Telnet a un host remoto) desde su domicilio, la dirección IP concreta utilizada por la computadora portátil no tiene tanta importancia. En particular, el usuario podría perfectamente funcionar con una dirección que el ISP que da servicio a su domicilio le asignara temporalmente a la computadora portátil. Ya hemos visto en la Sección 4.4 que DHCP ya proporciona esta funcionalidad.

- *¿Qué infraestructura cableada de soporte hay disponible?* En todos los escenarios anteriores, hemos supuesto implícitamente que existe una infraestructura fija a la que el usuario móvil puede conectarse: por ejemplo, la red del ISP que da servicio a su domicilio, la red de acceso inalámbrica de la oficina o las redes de acceso inalámbrico que atraviesan la autovía. ¿Qué sucede si no existe tal infraestructura? Si dos usuarios están próximos entre sí, desde el punto de vista de las comunicaciones, ¿pueden establecer una conexión de red en ausencia de cualquier otra infraestructura de la capa de red? Las redes ad hoc proporcionan precisamente estas capacidades. Este área en rápido desarrollo constituye la vanguardia de las investigaciones en redes móviles y cae fuera del alcance de este libro. En [Perkins 2000] y en las páginas web del grupo de trabajo de redes móviles ad hoc (manet) del IETF [manet 2009] se proporciona un tratamiento bastante exhaustivo de esta materia.

Para ilustrar los problemas implicados en el hecho de autorizar a un usuario móvil a mantener conexiones activas mientras se está desplazando entre varias redes, vamos a considerar una analogía humana. Un veinteañero que se va del domicilio familiar pasa a ser un adulto móvil, viviendo en una serie de habitaciones y/o apartamentos, cambiando a menudo de dirección. Si un viejo conocido quiere ponerse en contacto con él, ¿cómo puede ese conocido encontrar la dirección de su amigo móvil? Una forma común sería contactar con la familia, ya que un adulto móvil a menudo registrará ante la familia (le comunicará) su dirección actual (aunque sólo sea para que sus padres puedan enviarle dinero para pagarle el alquiler). El domicilio familiar, con su dirección permanente, se convertirá en ese lugar al que otros pueden acudir en primer lugar para poder comunicarse con ese adulto móvil. Las comunicaciones posteriores de ese viejo conocido pueden ser indirectas (por ejemplo, ese conocido puede enviar correo al domicilio de los padres y éstos reenviar el correo a nuestro adulto móvil) o directas (por ejemplo, cuando ese viejo conocido utiliza la dirección que le han facilitado los padres para enviar el correo directamente a su amigo móvil).

En un entorno de red, el domicilio permanente de un nodo móvil (como por ejemplo una computadora portátil o una PDA) se conoce con el nombre de **red propia** (*home network*) y la entidad dentro de la red propia que se encarga de llevar a cabo las funciones de gestión de la movilidad, de las que hablaremos más adelante, por cuenta del nodo móvil se conoce como **agente propio** (*home agent*). La red en la que reside actualmente el nodo móvil se conoce con el nombre de **red ajena** o **visitada** (*foreing or visited network*) y la entidad dentro de la red ajena que ayuda al nodo móvil con las funciones de gestión de la movilidad, que veremos más adelante, se conoce con el nombre de **agente ajeno** (*foreing agent*). Para los profesionales móviles, lo más probable es que su red propia sea la red de su empresa, mientras que la red ajena podría ser la de un colega al que estén visitando. Un **corresponsal** (*correspondent*) es la entidad que se quiere comunicar con el nodo móvil. La Figura 6.22 ilustra estos conceptos, así como los conceptos de direccionamiento considerados más adelante. En la Figura 6.22, observe que los agentes se muestran como si estuvieran ubicados en routers (por ejemplo, como procesos ejecutándose en routers), pero también podrían ejecutarse en otros hosts o servidores de la red.

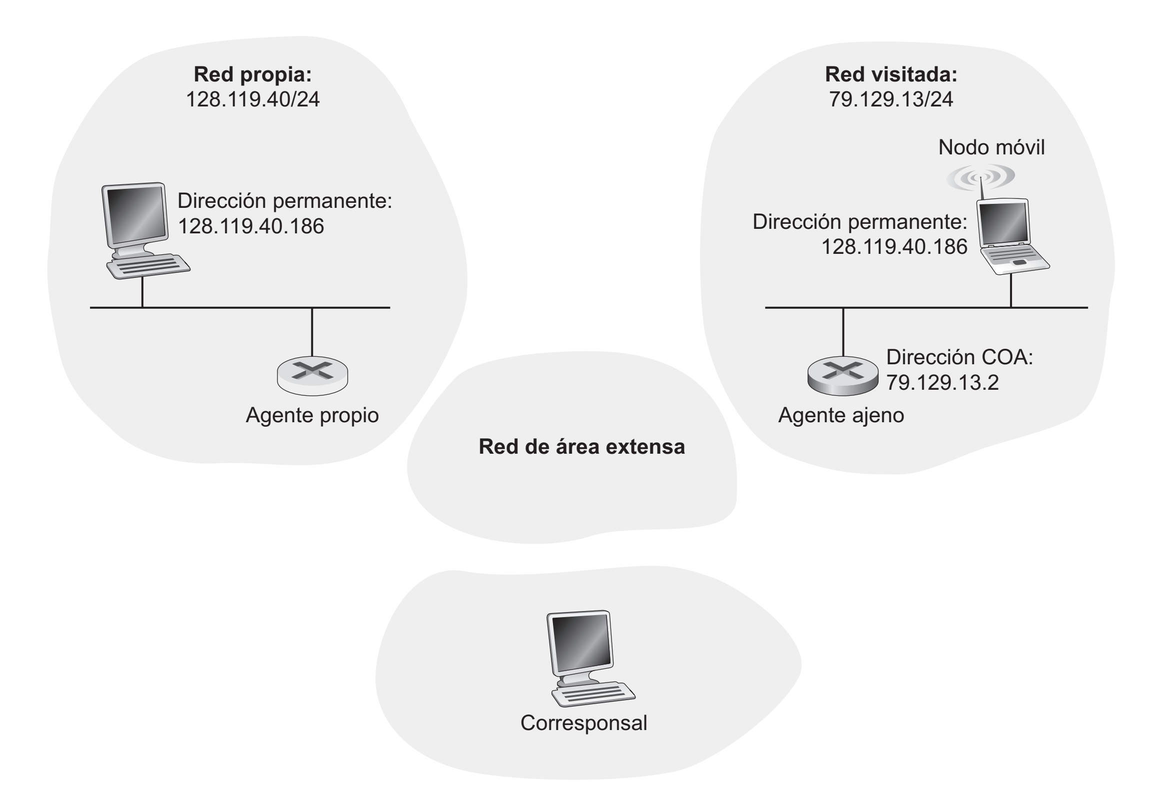


Figura 6.22 • Elementos iniciales de una arquitectura de red móvil.

6.5.1 Direccionamiento

Hemos observado anteriormente que, para que la movilidad de los usuarios sea transparente a ojos de las aplicaciones de red, es deseable que los nodos móviles conserven su dirección mientras se mueven de una red a otra. Cuando un nodo móvil reside en una red ajena, todo el tráfico dirigido a la dirección permanente de dicho nodo ahora tendrá que ser enrutado hacia la red ajena. ¿Cómo puede hacerse esto? Una opción es que la red ajena anuncie a todas las demás redes que el nodo móvil está residiendo ahora en su red. Esto podría hacerse mediante el intercambio usual de información de enrutamiento entre dominios y dentro de los dominios y requeriría pocos cambios en la infraestructura de enrutamiento existente. La red ajena podría simplemente anunciar a sus vecinos que dispone de una ruta altamente específica hacia la dirección permanente del nodo móvil (es decir, básicamente consistiría en informar a otras redes de que dispone de la ruta correcta para enviar datagramas a la dirección permanente del nodo móvil; véase la Sección 4.4). Esos vecinos propagarían entonces esa información de enrutamiento por toda la red como parte del procedimiento normal de actualización de la información de enrutamiento y de las tablas de reenvío. Cuando el nodo móvil abandone una red ajena y se una a otra, la nueva red ajena anunciará una nueva ruta altamente específica hacia el nodo móvil y la antigua red ajena retirará su información de enrutamiento relativa a ese nodo móvil.

Esto resuelve dos problemas a la vez, y lo hace sin llevar a cabo cambios significativos en la infraestructura de la capa de red. Otras redes conocerán la ubicación del nodo móvil y será fácil enrutar los datagramas hacia él, ya que las tablas de reenvío dirigirán los datagramas a la red ajena. Sin embargo, una desventaja significativa es la de la escalabilidad. Si la gestión de la movilidad tuviera que ser responsabilidad de los routers de la red, los routers tendrían que mantener entradas en sus tablas de reenvío para un número potencialmente muy alto de millones de nodos móviles y actualizar dichas entrada a medida que los nodos se movieran. En los problemas incluidos al final del capítulo se exploran otras desventajas adicionales.

Un enfoque alternativo (que además se ha adoptado en la práctica) consiste en trasladar la funcionalidad de movilidad desde el núcleo de la red hasta la frontera de la misma, lo cual es un tema recurrente en nuestro estudio de la arquitectura de Internet. Una forma natural de hacer esto es mediante la red propia del nodo móvil. De la misma forma que los padres del veinteañero móvil se encargan de controlar la ubicación de su hijo, el agente propio situado en la red propia del nodo móvil puede controlar en qué red ajena reside el nodo móvil. Obviamente, será necesario que exista un protocolo entre el nodo móvil (o un agente ajeno que represente al nodo móvil) y el agente propio para poder actualizar la ubicación del nodo móvil.

Consideremos ahora el agente ajeno con un poco más de detalle. El enfoque conceptualmente más simple, mostrado en la Figura 6.22, consiste en ubicar los agentes ajenos en los routers de frontera de la red ajena. Un papel del agente ajeno consiste en crear la denominada **dirección cedida (COA, *care-of address*)** para el nodo móvil, en la que la parte de red de la COA se corresponde con la de la red ajena. Por tanto, habrá dos direcciones asociadas con un nodo móvil, su **dirección permanente** (análoga a la dirección del domicilio familiar de nuestro veinteañero móvil) y su COA, a veces denominada **dirección ajena** (análoga a la dirección de la vivienda en la que está residiendo actualmente nuestro veinteañero móvil). En el ejemplo de la Figura 6.22, la dirección permanente del nodo móvil es 128.119.40.186. Cuando está visitando la red 79.129.13/24, el nodo móvil tiene una COA igual a 79.129.13.2. Un segundo papel del agente ajeno consiste en informar al agente propio de que el nodo móvil reside en su red (en la del agente ajeno) y tiene la COA indicada. Veremos enseguida que la COA se utiliza para "re-enrutar" datagramas hacia el nodo móvil a través de su agente ajeno.

Aunque hemos separado la funcionalidad del nodo móvil y del agente ajeno, merece la pena indicar que el nodo móvil también puede asumir las responsabilidades del agente ajeno. Por ejemplo, el nodo móvil podría obtener una COA en la red ajena (por ejemplo, utilizando un protocolo como DHCP) e informar él mismo al agente propio de cuál es su COA.

6.5.2 Enrutamiento hacia un nodo móvil

Hemos visto cómo un nodo móvil puede obtener una COA y cómo puede informar a su agente propio de dicha dirección. Pero hacer que el agente propio conozca la COA sólo resuelve parte del problema. ¿Cómo deberían direccionarse los datagramas y cómo deberían reenviarse hacia el nodo móvil? Puesto que sólo el agente propio (y no los routers del resto de la red) conoce la ubicación del nodo móvil, ya no bastará con dirigir simplemente un datagrama a la dirección permanente del nodo móvil y enviarlo hacia la infraestructura de la capa de red, sino que hay que hacer algo más. Podemos identificar dos enfoques distintos, a los que denominaremos enrutamiento indirecto y enrutamiento directo.

Enrutamiento indirecto hacia un nodo móvil

Consideremos en primer lugar un corresponsal que desea enviar un datagrama a un nodo móvil. Con la técnica del **enrutamiento indirecto**, el corresponsal simplemente direcciona el datagrama con la dirección permanente del nodo móvil y lo envía a la red, completamente ignorante de si el nodo móvil reside en su red propia o está visitando una red ajena; por tanto, la movilidad es completamente transparente para el corresponsal. Dichos datagramas se enrutan primero de la forma habitual hacia la red propia del nodo móvil. Esto se ilustra en el paso 1 de la Figura 6.23.

Volvamos ahora nuestra atención hacia el agente propio. Además de ser responsable de interactuar con un ajeno para saber en todo momento la COA del nodo móvil, el agente propio tiene otra función muy importante. Su segunda tarea consiste en estar atento para ver si llegan datagramas dirigidos a nodos cuya red propia sea la de ese agente propio, pero que estén actualmente residiendo en una red ajena. El agente propio intercepta estos datagramas y luego los reenvía hacia un nodo móvil siguiendo un proceso de dos pasos. Primero, el datagrama es reenviado hacia el agente ajeno, utilizando la dirección COA del nodo móvil (paso 2 de la Figura 6.23) y luego es reenviado desde el agente ajeno hacia el nodo móvil (paso 3 de la Figura 6.23).

Es bastante instructivo analizar este re-enrutamiento con mayor detalle. El agente propio necesitará direccionar el datagrama utilizando la COA del nodo móvil, de modo que la capa de red enruta el datagrama hacia la red ajena. Por otro lado, es deseable dejar el datagrama

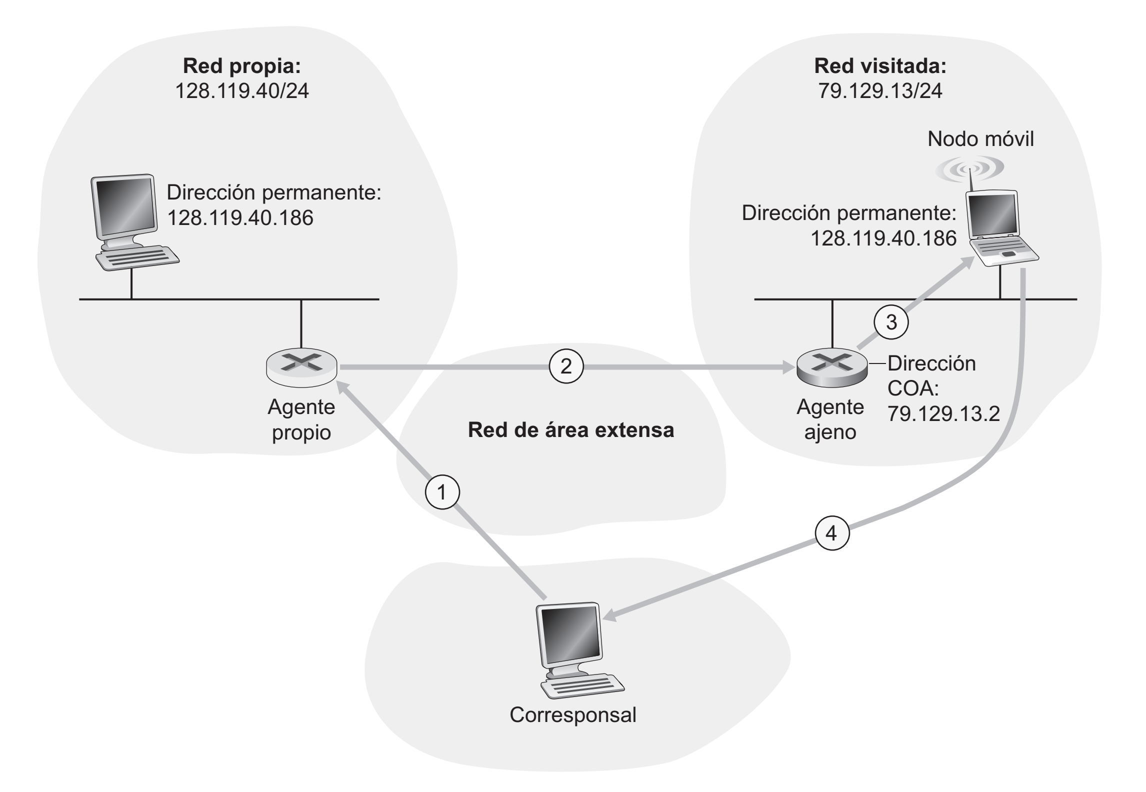


Figura 6.23 • Enrutamiento indirecto a un nodo móvil.

del corresponsal intacto, ya que la aplicación que va a recibir el datagrama no debe ser consciente de que el datagrama ha sido reenviado a través del agente propio. Ambos objetivos pueden satisfacerse haciendo que el agente propio **encapsule** el datagrama completo original del corresponsal dentro de un nuevo datagrama más grande. Este datagrama de mayor tamaño se direcciona y se entrega a la COA del nodo móvil. El agente ajeno, que "posee" la COA, recibirá y desencapsulará el datagrama; es decir, extraerá el datagrama original del corresponsal de dentro del datagrama más grande que lo encapsula y reenviará (paso 3 de la Figura 6.23) el datagrama original hacia el nodo móvil. La Figura 6.24 muestra un datagrama original de un corresponsal que es enviado hacia la red propia, un datagrama encapsulado enviado hacia el agente ajeno y el datagrama original entregado al nodo móvil. El lector atento observará que la encapsulación/desencapsulación aquí descrita es idéntica al concepto de tunelización, explicado en el Capítulo 4 en el contexto de la multidifusión IP y de IPv6.

Consideremos ahora cómo un nodo móvil envía datagramas a un corresponsal. Esto es bastante simple, ya que el nodo móvil puede dirigir sus datagramas *directamente* al corresponsal (utilizando su propia dirección permanente como la dirección de origen y la dirección del corresponsal como dirección de destino). Puesto que el nodo móvil conoce la dirección del corresponsal, no hay ninguna necesidad de enrutar el datagrama a través del agente propio. Esto se muestra en el paso 4 de la Figura 6.23.

Resumamos nuestras explicaciones acerca del enrutamiento indirecto indicando la nueva funcionalidad requerida en la capa de red para soportar la movilidad.

- *Un protocolo entre el nodo móvil y el agente ajeno.* El nodo móvil se registrará ante el agente ajeno cuando se conecte a la red ajena. De forma similar, el nodo móvil se desregistrará ante el agente ajeno cuando abandone la red ajena.

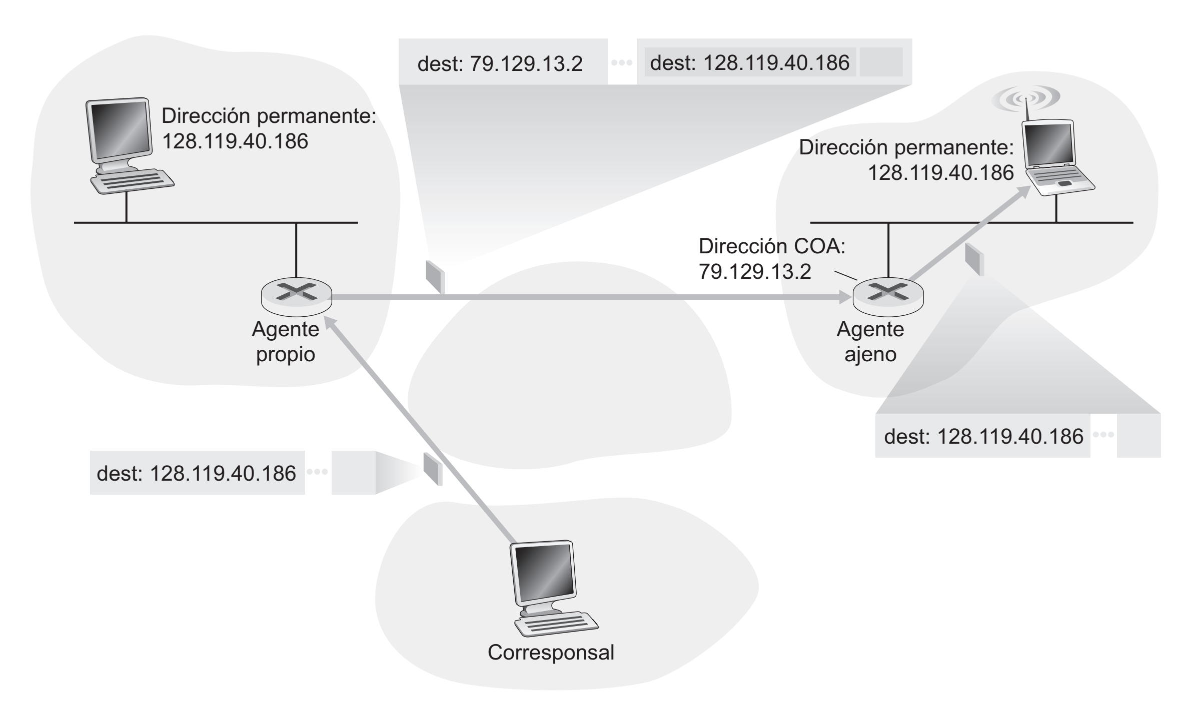


Figura 6.24 • Encapsulación y desencapsulación.

- *Un protocolo de registro entre el agente ajeno y el agente propio.* El agente ajeno registrará la COA del nodo móvil ante el agente propio. El agente ajeno no necesita desregistrar explícitamente una COA cuando un nodo móvil abandona su red, porque el subsiguiente registro de una nueva COA, cuando el nodo móvil se desplace a otra red, se encargará de ello.
- *Un protocolo de encapsulación de datagramas para el agente propio.* Este protocolo se encargará de la encapsulación y del reenvío del datagrama original del corresponsal dentro de un datagrama dirigido a la COA.
- *Un protocolo de desencapsulación para el agente ajeno.* Este protocolo se encargará de la extracción del datagrama original del corresponsal a partir del datagrama encapsulante y del reenvío del datagrama original al nodo móvil.

La exposición anterior proporciona todos los elementos (agentes ajenos, el agente propio y reenvío indirecto) necesarios para que un nodo móvil pueda mantener una conexión activa mientras se desplaza de una red a otra. Como ejemplo del modo en que estos elementos encajan, suponga que el nodo móvil está conectado a la red ajena A, que ha registrado una COA en la red A ante su agente propio y que está recibiendo datagramas que se están enrutando indirectamente a través de su agente propio. El nodo móvil se desplaza ahora a la red ajena B y se registra ante el agente ajeno de esa red, el cual informa al agente propio de la nueva COA del nodo móvil. A partir de ese momento, el agente propio re-enrutará los datagramas hacia la red ajena B. En lo que a un corresponsal respecta, la movilidad es transparente, ya que los datagramas se enrutan a través del mismo agente propio, tanto antes como después de que el nodo móvil se haya desplazado. En lo que respecta al agente propio, no existe ninguna interrupción en el flujo de datagramas, ya que los datagramas que vayan llegando se reenvían primero hacia la red ajena A y, después del cambio de COA, se reenvían a la red ajena B. ¿Pero qué pasa con el nodo móvil? ¿Experimentará una interrupción en el flujo de datagramas mientras se desplaza de una red a la otra? Mientras que el tiempo que transcurre entre la desconexión del nodo móvil de la red A (en cuyo momento ya no puede recibir datagramas a través de A) y la conexión a la red B (en cuyo momento registrará una nueva COA ante su agente propio) sea pequeño, sólo se perderán unos pocos datagramas. Recuerde del Capítulo 3 que las conexiones terminal a terminal pueden sufrir pérdidas de datagramas debidas a la congestión de la red. Por tanto, la pérdida ocasional de datagramas en una conexión cuando un nodo se desplaza de una red a otra no constituye en modo alguno un problema catastrófico. Si hiciera falta una comunicación libre de pérdidas, los mecanismos de la capa superior podrían recuperarse de la pérdida de datagramas, independientemente de si dicha pérdida es el resultado de una congestión en la red o de la movilidad del usuario.

En el estándar de IP móvil [RFC 3344], del que hablaremos en la Sección 6.6, se utiliza una técnica de enrutamiento indirecto.

Enrutamiento directo hacia un nodo móvil

La técnica basada en el enrutamiento indirecto ilustrada en la Figura 6.23 se ve aquejada de una ineficiencia conocida como el **problema del enrutamiento triangular**: los datagramas dirigidos al nodo móvil deben enrutarse en primer lugar hacia el agente propio y luego hacia la red ajena, aún cuando exista una ruta mucho más eficiente entre el corresponsal y el nodo móvil. En el caso peor, imagine un usuario móvil que está visitando la red ajena de un colega. Los dos están sentados uno al lado del otro e intercambiando datos a través de la red.

Los datagramas del corresponsal (en este caso, el colega del visitante) se enrutarán hacia el agente propio del usuario móvil y luego volverán a la red ajena.

La técnica de **enrutamiento directo** elimina la ineficiencia del enrutamiento triangular, pero el precio que hay que pagar es una mayor complejidad. Con la técnica del enrutamiento directo, un **agente corresponsal** situado en la red del corresponsal determina primero la COA del nodo móvil. Esto puede conseguirse haciendo que el agente corresponsal consulte al agente propio, suponiendo que (como en el caso del enrutamiento indirecto) el nodo móvil tiene registrado un valor actualizado de su COA ante el agente propio. También es posible que el propio corresponsal lleve a cabo la función del agente corresponsal, al igual que el nodo móvil puede realizar la función del agente ajeno. Esto se muestra en los pasos 1 y 2 de la Figura 6.25. Entonces, el agente corresponsal tuneliza los datagramas (pasos 3 y 4 de la Figura 6.25) directamente hacia la COA del nodo móvil, de forma análoga a la tunelización que lleva a cabo el agente propio.

Aunque el enrutamiento directo resuelve el problema del enrutamiento triangular, plantea dos problemas adicionales de importancia:

- Hace falta un ***protocolo de localización de usuarios móviles*** para que el agente corresponsal consulte al agente propio con el fin de obtener la COA del nodo móvil (pasos 1 y 2 de la Figura 6.25).

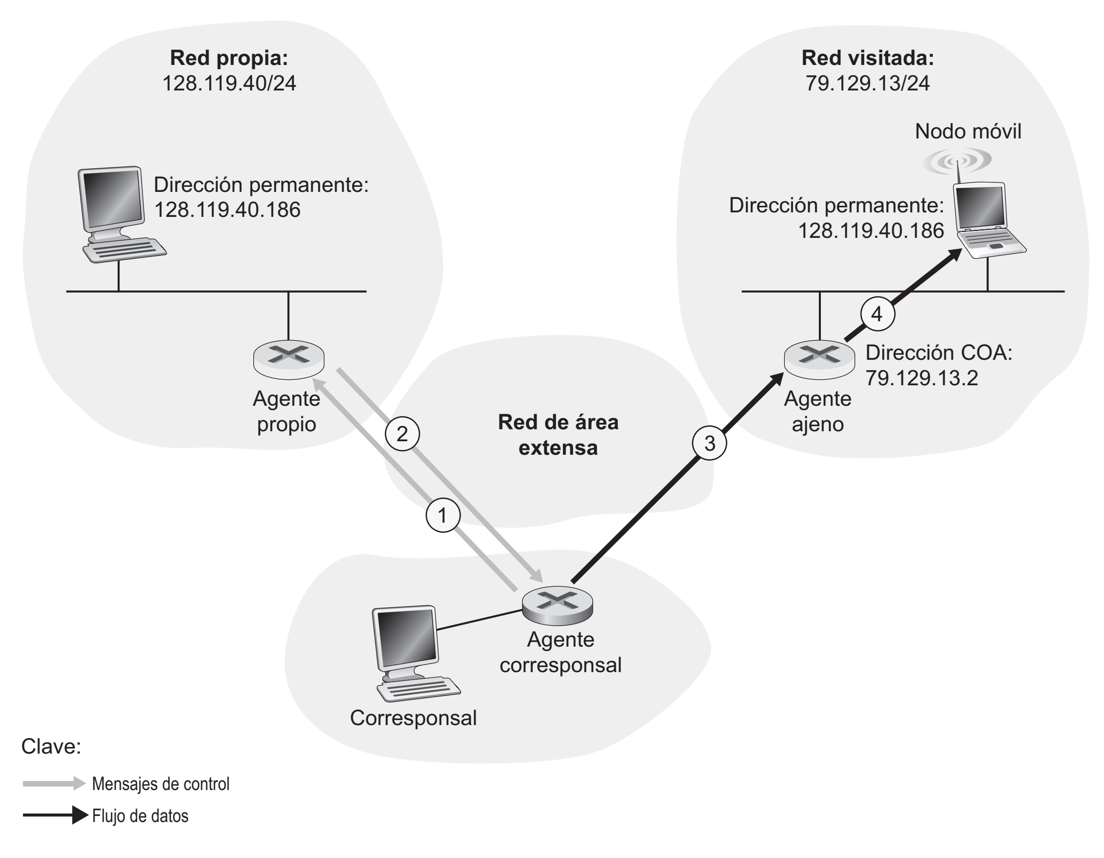


Figura 6.25 • Enrutamiento directo a un usuario móvil.

- Cuando el nodo móvil se desplaza de una red ajena a otra, ¿cómo podremos entonces reenviar los datos hacia la nueva red ajena? En el caso del enrutamiento indirecto, este problema se resolvía fácilmente actualizando la COA mantenida por el agente propio. Sin embargo, con el enrutamiento directo, el agente corresponsal sólo consulta la COA una vez al agente propio, al comienzo de la sesión. Por tanto, la actualización de la COA en el agente propio, aunque sigue siendo necesaria, no será suficiente para resolver el problema de cómo enrutar los datos hacia la nueva red ajena del nodo móvil.

Una solución sería crear un nuevo protocolo para notificar al corresponsal el cambio de la COA. Otra solución alternativa, que veremos adoptada en la práctica en las redes GSM, funciona de la forma siguiente: suponga que los datos están siendo reenviados actualmente hacia el nodo móvil en la red ajena en la que dicho nodo estaba ubicado en el momento de comenzar la sesión (paso 1 de la Figura 6.26). Identificaremos al agente ajeno de dicha red ajena en la que el nodo móvil se encontraba inicialmente con el nombre de **agente ajeno ancla**. Cuando el nodo móvil se desplaza a una nueva red ajena (paso 2 de la Figura 6.26) se registra ante el nuevo agente ajeno (paso 3) y el nuevo agente ajeno proporciona al agente ajeno ancla la nueva COA del nodo móvil (paso 4). Cuando el agente ajeno ancla recibe un datagrama encapsulado para un nodo móvil que ya ha salido de su red, puede entonces reencapsular el datagrama y reenviarlo al nodo móvil (paso 5) utilizando la nueva COA. Si el nodo móvil se desplaza posteriormente otra vez a otra nueva red ajena, el agente ajeno de esa nueva red visitada contactará con el agente ajeno ancla para establecer el reenvío hacia esta nueva red ajena.

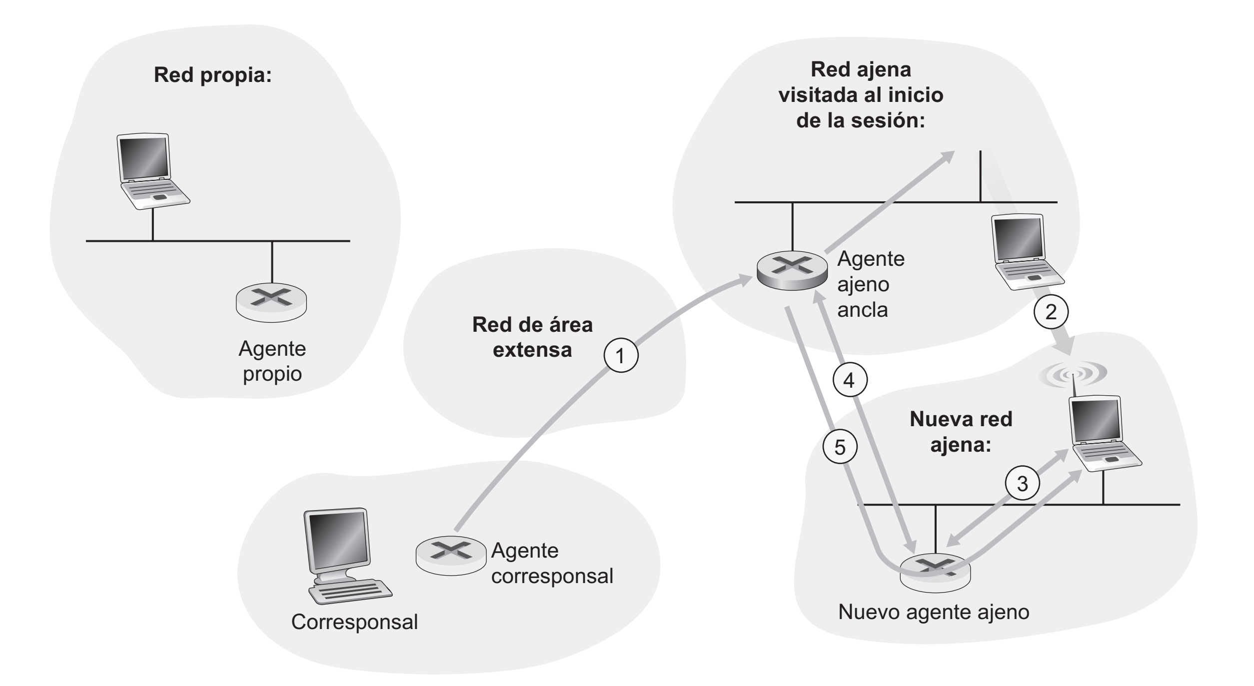


Figura 6.26 • Transferencia de móviles entre redes con enrutamiento directo.

6.6 IP móvil

Los protocolos y la arquitectura de Internet necesarios para dar soporte a la movilidad, colectivamente conocidos con el nombre de IP móvil, están definidos principalmente en RFC 3344 para IPv4. IP móvil es un estándar flexible, que soporta muchos modos distintos de operación (por ejemplo, operación con o sin un agente ajeno), múltiples formas de que los agentes y los nodos móviles se descubran entre sí, utilización de direcciones COA únicas o múltiples y diversas formas de encapsulación. Como tal, IP móvil es un estándar complejo y haría falta un libro completo para describirlo en detalle; de hecho, uno de tales libros es [Perkins 1998b]. Nuestro objetivo aquí, mucho más modesto, es proporcionar una panorámica de los aspectos más importantes de IP móvil e ilustrar su uso en unos cuantos escenarios comunes.

La arquitectura de IP móvil contiene muchos de los elementos que hemos presentado anteriormente, incluyendo los conceptos de agente propio, agente ajeno, direcciones COA y encapsulación/desencapsulación. El estándar actual [RFC 3344] especifica el uso del enrutamiento indirecto hacia el nodo móvil.

El estándar IP móvil consta de tres elementos principales:

- *Descubrimiento de agentes.* IP móvil define los protocolos utilizados por un agente propio o ajeno para anunciar sus servicios a los nodos móviles, así como protocolos para que los nodos móviles soliciten los servicios de un agente ajeno o propio.
- *Registro ante el agente propio.* IP móvil define los protocolos utilizados por el nodo móvil y/o el agente ajeno para registrar y desregistrar direcciones COA ante el agente propio de un nodo móvil.
- *Enrutamiento indirecto de los datagramas.* El estándar también define la forma en que el agente propio reenvía los datagramas hacia los nodos móviles, incluyendo reglas para el reenvío de datagramas, reglas para la gestión de las condiciones de error y diversas formas de encapsulación [RFC 2003, RFC 2004].

Las consideraciones de seguridad tienen una gran importancia a lo largo de todo el estándar de IP móvil. Por ejemplo, es clara la necesidad de que un nodo móvil se autentique para garantizar que los usuarios maliciosos no puedan registrar una dirección COA falsa ante un agente propio que haría que todos los datagramas destinados a una dirección IP fueran redirigidos hacia el usuario malicioso. En IP móvil la seguridad se consigue utilizando muchos de los mecanismos que examinaremos en el Capítulo 8, por lo que en las explicaciones que siguen no vamos a tocar los problemas de seguridad.

Descubrimiento de agentes

Un nodo de IP móvil que llegue a una nueva red, independientemente de si está conectando a una red ajena o volviendo a la suya propia, debe averiguar la identidad del agente ajeno o propio correspondiente. De hecho, es el descubrimiento de un nuevo agente ajeno, con una nueva dirección de red, el que permite a la capa de red del nodo móvil averiguar que acaba de entrar en una nueva red ajena. Este proceso se conoce con el nombre de **descubrimiento de agentes**. El descubrimiento de agentes puede realizarse de una de dos formas: mediante los anuncios de los agentes o mediante las solicitudes de agente.

Con el **anuncio de agente**, cada agente ajeno o propio anuncia sus servicios, utilizando una extensión del protocolo existente de descubrimiento de router [RFC 1256]. El agente difunde periódicamente un mensaje ICMP con un campo de tipo de valor 9 (descubrimiento de router) a través de todos los enlaces con los que esté conectado. El mensaje de descubrimiento de router contiene la dirección IP del router (es decir, el agente), permitiendo así que los nodos móviles conozcan la dirección IP del agente. El mensaje de descubrimiento de router también contiene una extensión de anuncio de agente de movilidad que contiene información adicional que el nodo móvil necesita. Entre los campos más importantes de esa extensión se encuentran los siguientes:

- *Bit de agente propio (H).* Indica que el agente es un agente propio para la red en la que reside.
- *Bit de agente ajeno (F).* Indica que el agente es un agente ajeno para la red en la que reside.
- *Bit de registro requerido (R).* Indica que los usuarios móviles en esta red *deberán* registrarse ante un agente ajeno. En particular, un usuario móvil no podrá obtener una dirección COA en la red ajena (por ejemplo, utilizando DHCP) y asumir él mismo la funcionalidad del agente ajeno sin antes registrarse ante el agente ajeno.
- *Bits de encapsulación M, G.* Indican si se utilizará algún tipo de encapsulación distinta de la encapsulación IP-en-IP.
- *Campos de dirección COA.* Es una lista de una o más direcciones COA proporcionada por el agente ajeno. En el ejemplo que indicamos más adelante, la dirección COA estará asociada con el agente ajeno, que recibirá los datagramas destinados a esa COA y luego los reenviará al nodo móvil apropiado. El usuario móvil seleccionará una de estas direcciones como dirección COA a la hora de registrarse ante su agente propio.

La Figura 6.27 ilustra algunos de los campos clave contenidos en el mensaje de anuncio de agente.

Con la **solicitud de agente**, un nodo móvil que quiera averiguar información acerca de los agentes sin esperar a recibir un anuncio de agente puede difundir un mensaje de solicitud de agente, que es simplemente un mensaje ICMP con un campo de tipo que contiene el valor 10. Un agente que reciba la solicitud enviará directamente al nodo móvil un anuncio de agente, mediante un mensaje de unidifusión, pudiendo entonces el nodo móvil proceder como si hubiera recibido un anuncio no solicitado.

Registro ante el agente propio

Una vez que un nodo IP móvil ha recibido una dirección COA, debe registrar dicha dirección ante su agente propio. Esto puede hacerse a través del agente ajeno (que se encargará de registrar la COA ante el agente propio) o lo puede hacer también directamente el propio nodo de IP móvil. Vamos a analizar el primero de los casos, que consta de cuatro pasos:

1. Después de recibir el anuncio de un agente ajeno, el nodo móvil envía un mensaje de registro de IP móvil a ese agente ajeno. El mensaje de registro se transporta dentro de un datagrama UDP y se envía al puerto 434. El mensaje de registro incluye una dirección COA anunciada por el agente ajeno, además de la dirección del agente propio (HA), la dirección permanente del nodo móvil (MA), el tiempo de vida solicitado para

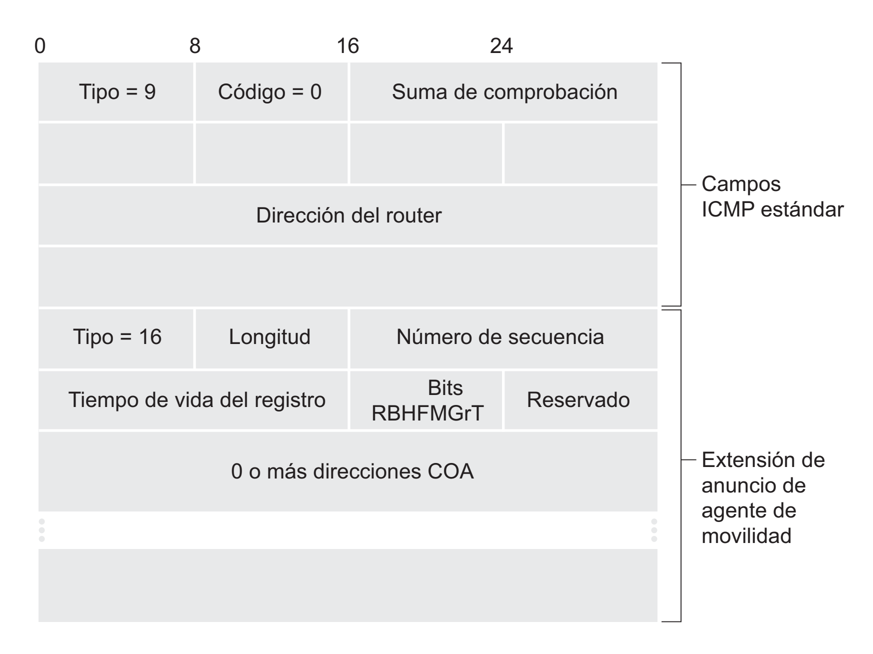


Figura 6.27 • Mensaje de descubrimiento de router ICMP con la extensión de anuncio de agente de movilidad.

el registro y un identificador de registro de 64 bits. El tiempo de vida solicitado para el registro es el número de segundos durante los cuales el registro tendrá validez. Si el registro no se renueva ante el agente propio dentro de ese tiempo de vida especificado el registro quedará invalidado. El identificador de registro actúa como un número de secuencia y sirve para establecer la correspondencia entre una respuesta de registro recibida y una solicitud de registro, como se indica más adelante.

2. El agente ajeno recibe el mensaje de registro y anota la dirección IP permanente del nodo móvil. El agente ajeno ahora sabe que debe buscar datagramas que contengan un datagrama encapsulado cuya dirección de destino coincida con esa dirección permanente del nodo móvil. El agente ajeno envía a continuación un mensaje de registro de IP móvil (de nuevo dentro de un datagrama UDP) al puerto 434 del agente propio. El mensaje contiene las direcciones COA, HA y MA, el formato de encapsulación solicitado, el tiempo de vida de registro solicitado y el identificador de registro.
3. El agente propio recibe la solicitud de registro y comprueba la autenticidad y la corrección de la misma. El agente propio establece una asociación entre la dirección IP permanente del nodo móvil y la dirección COA. De este modo, en el futuro, los datagramas que el agente propio reciba y que estén dirigidos al nodo móvil serán encapsulados y tunelizados hacia la dirección COA. El agente propio envía una respuesta de registro de IP móvil que contiene las direcciones HA y MA, el tiempo de vida real del registro y el identificador de registro correspondiente a la solicitud que se esté satisfaciendo con esta respuesta.
4. El agente ajeno recibe la respuesta de registro y a continuación la reenvía hacia el nodo móvil.

Llegados a este punto, el proceso de registro estará terminado y el nodo móvil podrá recibir los datagramas enviados hacia su dirección permanente. La Figura 6.28 ilustra estos pasos. Observe que el agente propio especifica un tiempo de vida más pequeño que el tiempo de vida solicitado por el nodo móvil.

Un agente ajeno no necesita desregistrar explícitamente una dirección COA cuando un nodo móvil abandone su red. Ese proceso de cancelación de la dirección tendrá lugar automáticamente cuando el nodo móvil se desplace a una red nueva (independientemente de si es otra red ajena o si se trata de su red propia) y registre una nueva dirección COA.

El estándar de IP móvil permite muchas capacidades y escenarios adicionales a los que hemos descrito. El lector interesado puede consultar [Perkins 1998b; RFC 3344].

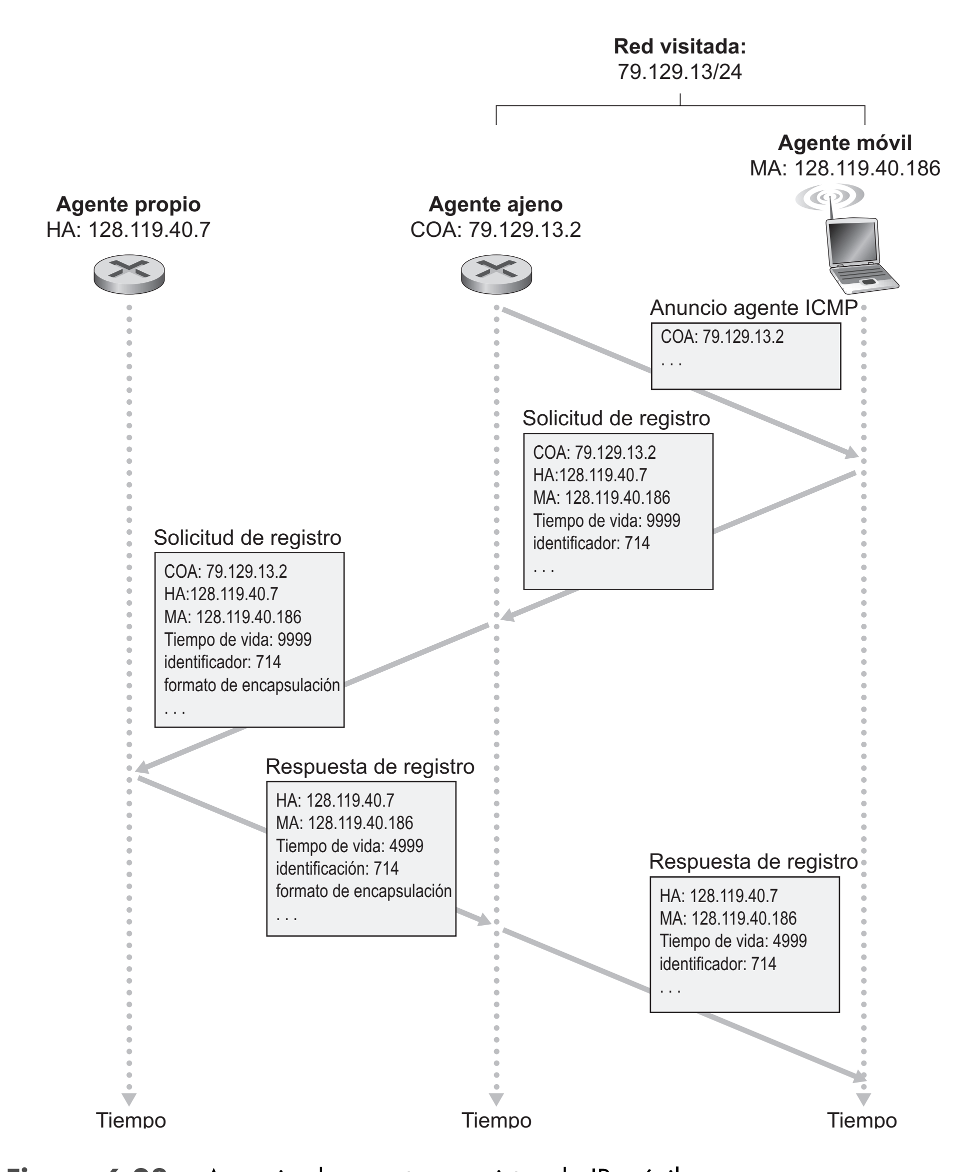


Figura 6.28 • Anuncio de agente y registro de IP móvil.

6.7 Gestión de la movilidad en redes celulares

Habiendo examinado cómo se gestiona la movilidad en las redes IP, volvamos ahora nuestra atención a otras redes que tienen una historia mucho más dilatada de soporte de la movilidad: las redes de telefonía celular. Mientras que en la Sección 6.4 nos hemos centrado en el enlace inalámbrico del primer salto dentro de las redes celulares, aquí vamos a centrarnos en la movilidad, utilizando como caso de estudio la arquitectura de las redes celulares GSM [Goodman 1997; Mouly 1992; Scourias 1997; Kaaranen 2001; Korhonen 2003; Turner 2009], dado que se trata de una tecnología madura y de amplia implantación. Como en el caso de IP móvil, veremos que la arquitectura de las redes GSM integra varios de los principios fundamentales que hemos identificado en la Sección 6.5.

Al igual que IP móvil, GSM adopta una técnica basada en el enrutamiento indirecto (véase la Sección 6.5.2), enrutando primero la llamada del corresponsal hacia la red propia del usuario móvil y de allí a la red visitada. En terminología GSM, la red propia del usuario móvil se denomina **red móvil terrestre pública propia (home PLMN, *home Public Land Mobile Network*)**. Dado que el acrónimo PLMN resulta un tanto complicado de pronunciar y dado que tratamos de evitar una sopa de letras de acrónimos, en lo sucesivo nos referiremos a la PLMN propia de GSM simplemente con el nombre de **red propia**. La red propia es el proveedor de telefonía celular con el que está abonado el usuario móvil (es decir, el proveedor que factura al usuario por los servicios mensuales de telefonía celular). La PLMN visitada, a la que denominaremos sencillamente **red visitada**, es la red en la que reside actualmente el usuario móvil.

Como en el caso de IP móvil, las responsabilidades de las redes propia y visitada son bastante distintas.

- La red propia mantiene una base de datos que se conoce con el nombre de **Registro de ubicaciones propias (HLR, *Home Location Register*)**, que contiene el número de teléfono celular permanente y la información del perfil de abonado para cada uno de sus abonados. Es importante resaltar que la base de datos HLR también contiene información acerca de las ubicaciones actuales de estos abonados. Es decir, si un usuario móvil está actualmente en situación de itinerancia (*roaming*) dentro de la red celular de otro proveedor, la HLR contendrá suficiente información como para obtener (a través de un proceso que vamos a describir en breve) una dirección, dentro de la red visitada, hacia la que enrutar las llamadas dirigidas al usuario móvil. Como veremos, un conmutador especial dentro de la red propia, conocido como **Centro de conmutación pasarela para servicios móviles (GMSC, *Gateway Mobile services Switching Center*)** es contactado por el corresponsal cada vez que realiza una llamada a un usuario móvil. De nuevo, para tratar de evitar el uso excesivo de acrónimos, en lo sucesivo nos vamos a referir al GMSC con el término más descriptivo de **MSC propio**.
- La red visitada mantiene una base de datos conocida con el nombre de **Registro de ubicación de visitantes (VLR, *Visitor Location Register*)**. La base de datos VLR contiene una entrada para cada usuario móvil que se encuentra *actualmente* en la parte de la red a la que da servicio VLR. Las entradas de VLR aparecen y desaparecen, por tanto, a medida que los usuarios móviles entran y salen de la red. La base de datos VLR normalmente está co-ubicada con el centro de conmutación móvil (MSC) que coordina el establecimiento de llamadas hacia y desde la red visitada.

En la práctica, la red celular de un proveedor sirve como red propia para sus abonados y como red visitada para los usuarios móviles que estén abonados a un proveedor diferente de telefonía celular.

6.7.1 Enrutamiento de llamadas hacia un usuario móvil

Ahora podemos describir cómo se realiza una llamada a un usuario móvil GSM que se encuentra en una red ajena. A continuación proporcionamos un ejemplo simple; el lector interesado puede encontrar otros escenarios más complejos en [Mouly 1992]. Los pasos, ilustrados en la Figura 6.29, son los siguientes:

1. El corresponsal marca el número telefónico del usuario móvil. Este número no hace referencia por sí mismo a ninguna línea telefónica o ubicación concreta (después de todo, el número telefónico es fijo, mientras que el usuario es móvil). Los primeros dígitos del número son suficientes para identificar globalmente la red propia a la que el móvil pertenece. La llamada será enrutada desde el corresponsal, a través de la red telefónica conmutada pública (PSTN, *Public Switched Telephone Network*), hasta el MSC propio de la red propia del móvil. Éste es el primer tramo de la llamada.
2. El MSC propio recibe la llamada e interroga a HLR para determinar la ubicación del usuario móvil. En el caso más simple, HLR devuelve el **Número de itinerancia de la estación móvil (MSRN, *Mobile Station Roaming Number*)**, al que en lo sucesivo denominaremos simplemente **número de itinerancia**. Observe que este número es diferente del número telefónico permanente del móvil que está asociado con la red propia del móvil. El número de itinerancia es efímero: es asignado tmporalmente al móvil cuando entra dentro de una red visitada. El número de itinerancia juega un papel similar al de la dirección COA en IP móvil y, al igual que la dirección COA, es invisible para el corresponsal y para el móvil. Si el registro HLR no tiene el número de itinerancia, devuelve la dirección del VLR en la red visitada. En este caso (no mostrado en la Figura 6.29), el MSC propio necesitará consultar a ese VLR para obtener el número de itinerancia del nodo móvil. Pero, ¿cómo obtiene el HLR el número de itinerancia o la dirección del VLR? ¿Qué sucede con estos valores cuando el usuario móvil se desplaza a otra red visitada? Analizaremos estas importantes cuestiones enseguida.
3. Conocido el número de itinerancia, el MSC propio establece el segundo tramo de la llamada a través de la red hasta el MSC de la red visitada. Con ello, la llamada se habrá completado, produciéndose el enrutamiento desde el corresponsal hasta el MSC propio, de éste al MSC visitado y de ahí a la estación base que da servicio al usuario móvil.

Una cuestión no resuelta en el paso 2 es cómo el HLR obtiene la información acerca de la ubicación del usuario móvil. Cuando se enciende un teléfono móvil o éste entra en una parte de una red visitada que está cubierta por un nuevo VLR, el móvil debe registrarse ante la red visitada. Esto se realiza intercambiando mensajes de señalización entre el móvil y el VLR. El VLR visitado, a su vez, envía un mensaje de solicitud de actualización de la ubicación al HLR del móvil. Este mensaje informa al HLR o bien del número de itinerancia a través del cual se puede contactar con el móvil, o de la dirección del VLR (al que se puede consultar más adelante para obtener el número de itinerancia del móvil). Como parte de este intercambio, el VLR obtiene también información de abonado de la base de datos HLR acerca del móvil y determina qué servicios (en su caso) tiene que proporcionar la red visitada al usuario móvil.

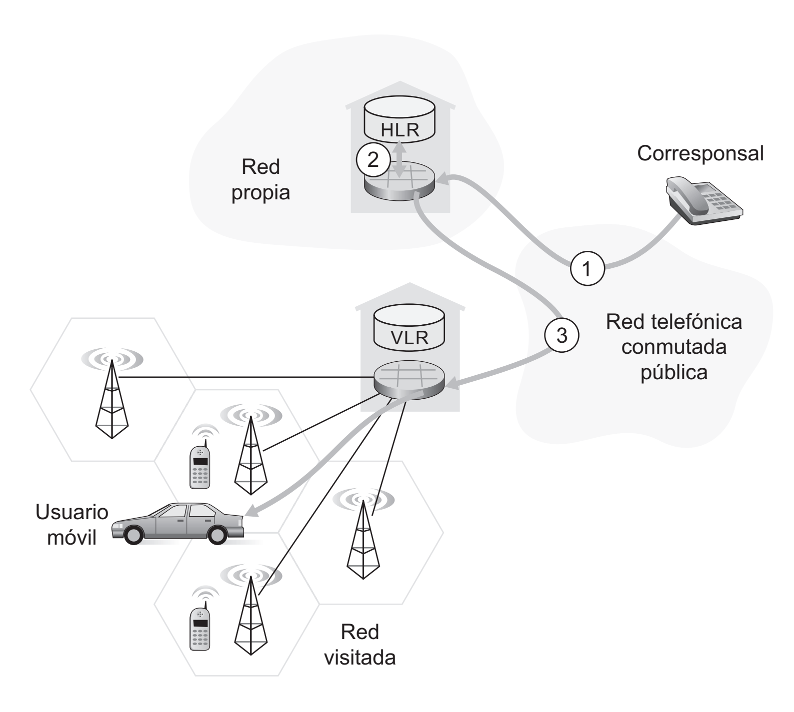


Figura 6.29 • Establecimiento de una llamada hacia un usuario móvil: enrutamiento indirecto.

6.7.2 Transferencia de llamadas en GSM

Cuando una estación móvil cambia su asociación de una estación base a otra durante una llamada se produce lo que se denomina **transferencia** o cesión de la llamada. Como se muestra en la Figura 6.30, la llamada del móvil es enrutada inicialmente (antes de la transferencia) hacia el móvil a través de una estación base (a la que denominaremos estación base antigua) mientras que después de la transferencia se enruta hacia el móvil a través de otra estación base (a la que denominaremos nueva estación base). Observe que una transferencia

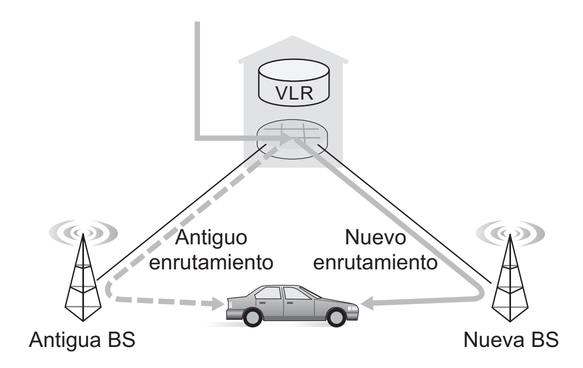


Figura 6.30 • Escenario de transferencia de llamadas entre estaciones base con un MSC común.

de llamada entre las estaciones base da como resultado no sólo que el móvil transmita/reciba hacia/desde una estación base nueva, sino también que la llamada activa se re-enrute desde un punto de conmutación dentro de la red hacia la nueva estación base. Inicialmente, vamos a suponer que las estaciones base nueva y antigua comparten el mismo MSC y que dicho re-enrutamiento tiene lugar en ese MSC.

Puede haber varias razones para que se produzca la transferencia de la llamada, entre las que podemos citar (1) que la señal entre la estación base actual y el móvil se puede haber deteriorado hasta tal punto que exista riesgo de que la llamada se interrumpa y (2) que una celda pueda haberse sobrecargado debido a que está gestionando un gran número de llamadas. Esta congestión puede aliviarse transfiriendo usuarios móviles a otras celdas cercanas menos congestionadas.

Mientras que está asociado con una estación base, el móvil mide periódicamente la intensidad de una señal baliza recibida desde su estación base actual, así como señales baliza procedentes de estaciones base cercanas que el móvil pueda "escuchar". Estas medidas son reenviadas una o dos veces por segundo hacia la estación base actual del móvil. Es la propia estación base antigua la que inicia la transferencia de una llamada en GSM basándose en estas medidas, en la carga actual de móviles existente en las celdas cercanas y en otros factores [Mouly 1992]. El estándar GSM no especifica el algoritmo concreto que una estación base debe utilizar a la hora de determinar si transferir o no una llamada.

La Figura 6.31 ilustra los pasos que se llevan a cabo cuando una estación base decide transferir un usuario móvil:

1. La estación base (BS) antigua informa al MSC visitado de que hay que realizar una transferencia, así como de la estación base (o del posible conjunto de estaciones base) a la que hay que transferir el móvil.
2. El MSC visitado inicia las operaciones de establecimiento de la ruta hacia la nueva estación base, asignando los recursos necesarios para transportar la llamada re-enrutada y señalizando a la nueva estación base que se va a producir una transferencia.
3. La nueva estación base asigna y activa un nuevo canal de radio para que lo utilice el móvil.

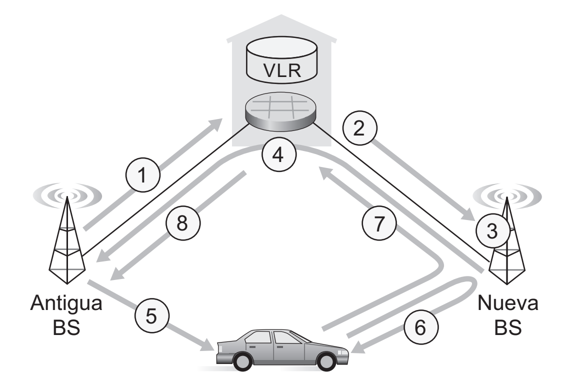


Figura 6.31 • Pasos necesarios para realizar una transferencia entre estaciones base con un MSC común.

4. La nueva estación base envía de vuelta al MSC visitado y a la estación base antigua una señal indicativa de que se ha establecido la ruta entre el MSC visitado y la nueva estación base, y de que hay que informar al móvil de la transferencia que se va a producir. La nueva estación base proporciona toda la información que el móvil necesitará para asociarse con ella.
5. El móvil es informado de que debe realizar una transferencia. Observe que hasta este punto, el móvil ha estado completamente ignorante de que la red estaba realizando el trabajo necesario para llevar a cabo una transferencia (por ejemplo, asignar un canal en la nueva estación base y asignar una ruta entre el MSC visitado y la nueva estación base).
6. El móvil y la nueva estación base intercambian uno o más mensajes para activar completamente el nuevo canal en la estación base nueva.
7. El móvil envía a la nueva estación base un mensaje indicando que se ha completado la transferencia, el cual es reenviado hacia el MSC visitado. Entonces, el MSC visitado re-enruta la llamada activa hacia el móvil a través de la nueva estación base.
8. Por último, se liberan los recursos asignados en la ruta que llevaba hacia la antigua estación base.

Vamos a concluir nuestro análisis de transferencia de llamadas considerando lo que ocurre cuando el móvil se desplaza a una estación base que está asociada con un MSC *diferente* al de la antigua estación base, y también lo que sucede cuando este tipo de transferencia entre dos MSC tiene lugar más de una vez. Como se muestra en la Figura 6.32, GSM define

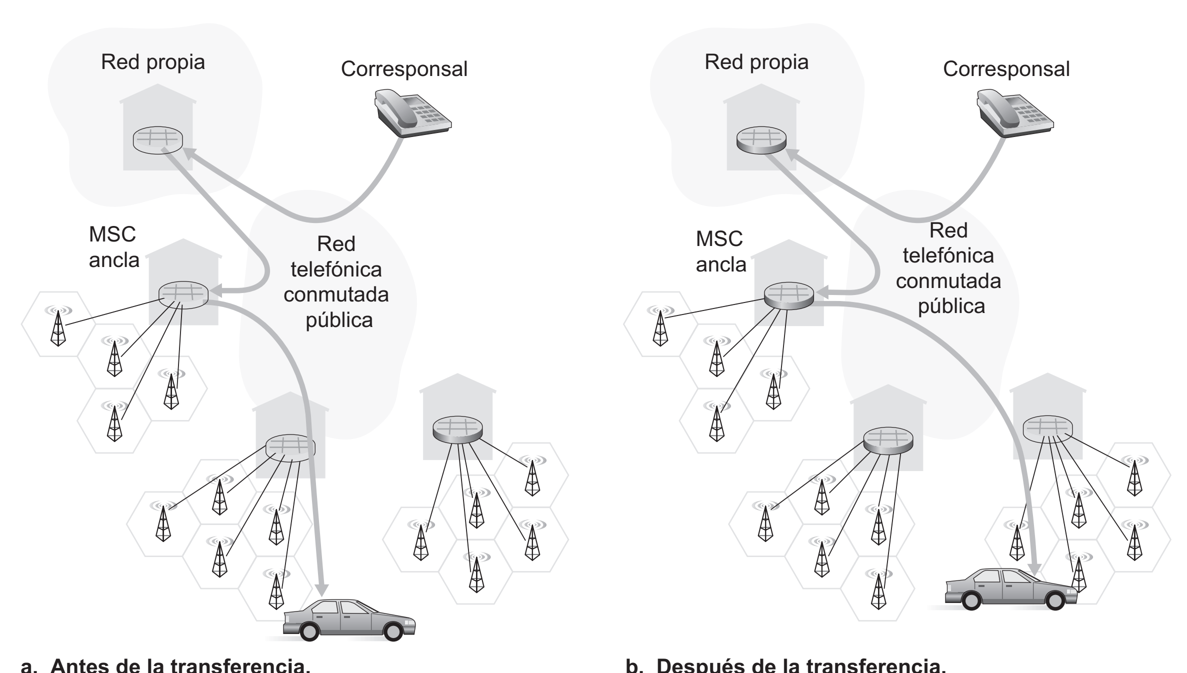


Figura 6.32 • Re-enrutamiento a través de un MSC ancla.

el concepto de **MSC ancla**. El MSC ancla es el MSC que estaba siendo visitado por el móvil cuando da comienzo una llamada; por tanto, el MSC ancla sigue siendo el mismo a todo lo largo de la llamada. Mientras dure la llamada e independientemente del número de transferencias inter-MSC realizadas por el móvil, la llamada es enrutada desde el MSC propio hasta el MSC ancla y después desde el MSC ancla al MSC visitado en el que el móvil esté situado actualmente. Cuando un móvil se desplaza desde el área de cobertura de un MSC a la de otro, la llamada activa se re-enruta desde el MSC ancla hasta el nuevo MSC visitado que contiene a la nueva estación base. Por tanto, en cada momento existen, como mucho, tres MSC (el MSC propio, el MSC ancla y el MSC visitado) entre el corresponsal y el móvil. La Figura 6.32 ilustra el enrutamiento de una llamada entre los MSC visitados por un usuario móvil.

En lugar de mantener un único salto de MSC desde el MSC ancla al MSC actual, una técnica alternativa habría sido encadenar simplemente los MSC visitados por el móvil, haciendo que cada MSC antiguo reenvíe la llamada activa al nuevo MSC cada vez que el móvil se desplace hasta un nuevo MSC. Dicho encadenamiento de centros MSC puede tener lugar en las redes celulares IS-41, con un paso opcional de minimización de la ruta para eliminar los MSC existentes entre el MSC ancla y el MSC visitado actual [Lin 2001].

Terminamos esta exposición acerca de la gestión de la movilidad GSM con una comparación de la gestión de movilidad en GSM e IP móvil. La comparación en la Tabla 6.2 indica que aunque las redes celulares e IP son fundamentalmente distintos en muchos aspectos, comparten un sorprendente número de elementos funcionales comunes y de técnicas globales a la hora de gestionar la movilidad.

Elemento GSM	Comentarios sobre el elemento GSM	Elemento IP móvil
Sistema propio	Red a la que pertenece el número de teléfono permanente del usuario móvil.	Red propia
Centro de conmutación móvil pasarela o simplemente MSC propio, Registro de ubicación propio (HLR)	MSC propio: punto de contacto para obtener la dirección enrutable del usuario móvil. HLR: base de datos en el sistema propio que contiene el número de teléfono permanente, la información de perfil, la ubicación actual del usuario móvil y la información de abono.	Agente propio
Sistema visitado	Red distinta del sistema propio en la que reside actualmente el usuario móvil.	Red visitada
Centro de conmutación de servicios móviles visitados, Registro de ubicación de visitantes (VLR)	MSC visitado: responsable de establecer las llamadas hacia/desde los nodos móviles en las celdas asociadas con el MSC. VLR: entrada temporal en la base de datos dentro del sistema visitado, que contiene información de subscripción para cada usuario móvil visitante.	Agente ajeno
Número de itinerancia de la estación móvil (MSRN) o simplemente número de itinerancia	Dirección enrutable para el segmento de llamada telefónica entre el MSC propio y el MSC visitado, que no es visible ni para el móvil ni para el corresponsal.	Dirección COA

Tabla 6.2 • Aspectos comunes entre IP móvil y GSM en lo que respecta a la movilidad.

6.8 Tecnología inalámbrica y movilidad: impacto sobre los protocolos de las capas superiores

En este capítulo, hemos visto que las redes inalámbricas difieren significativamente de sus contrapartidas cableadas tanto en la capa de enlace (como resultado de características del canal inalámbrico tales como el desvanecimiento, el multicamino y los terminales ocultos) como en la capa de red (como resultado de la existencia de usuarios móviles que cambian sus puntos de asociación con la red). Pero, ¿existen diferencias importantes en las capas de transporte y de aplicación? Resulta tentador pensar que estas diferencias serán menores, dado que la capa de red proporciona a las capas superiores el modelo de servicio de entrega de mejor esfuerzo tanto en las redes cableadas como en las inalámbricas. De forma similar, si se utilizan protocolos como TCP o UDP para proporcionar los servicios de la capa de transporte a las aplicaciones tanto en redes inalámbricas como cableadas, entonces la capa de aplicación tampoco tendría por qué ser modificada. En un cierto sentido, nuestra intuición es correcta: TCP y UDP pueden (y de hecho lo hacen) operar en redes con enlaces inalámbricos. Por otra parte, los protocolos de transporte en general, y TCP en particular, pueden tener en ocasiones un rendimiento muy distinto en las redes cableadas e inalámbricas, y es aquí, en términos de rendimiento, que las diferencias se manifiestan. Veamos por qué.

Recuerde que TCP retransmite los segmentos perdidos o corruptos a lo largo de la ruta existente entre el emisor y el receptor. En el caso de usuarios móviles, las pérdidas pueden ser el resultado de la congestión de la red (desbordamiento de los buffers de los routers) o de la transferencia de llamadas (por ejemplo, debido a retardos en el re-enrutamiento de segmentos hacia el nuevo punto de asociación con la red de un móvil). En todos los casos, los mensajes ACK del receptor al emisor de TCP sólo indican que un segmento no ha sido recibido intacto; el emisor no es consciente de si el segmento se ha perdido a causa de la congestión, durante la transferencia o porque se han detectado bits erróneos. En todos los casos, la respuesta del emisor es la misma: retransmitir el segmento. La respuesta del control de congestión de TCP es *también* siempre la misma en todos los casos: TCP disminuye el tamaño de su ventana de congestión como se explica en la Sección 3.7. Reduciendo incondicionalmente su ventana de congestión, TCP asume implícitamente que las pérdidas de segmento se deben a la congestión más que a la corrupción o a la transferencia de llamadas. Pero, como hemos visto en la Sección 6.2, los errores de bit son mucho más comunes en las redes inalámbricas que en las redes cableadas. Cuando se producen tales errores de bits o cuando tienen lugar pérdidas debido al mecanismo de transferencia no existe realmente ninguna razón para que el emisor TCP reduzca su ventana de congestión (y por tanto su velocidad de envío). De hecho, podría darse perfectamente el caso de que los buffers de los routers estén vacíos y de que los paquetes estén fluyendo a lo largo de la ruta terminal a terminal sin que la congestión represente ningún obstáculo.

Los investigadores se dieron cuenta a principios y mediados de la década de 1990 de que dadas las tasas de errores de bit en los enlaces inalámbricos y la posibilidad de que se produzcan pérdidas en la transferencia de llamadas, la respuesta del control de congestión de TCP podría ser problemática en una configuración inalámbrica. Para resolver este problema se pueden usar tres clases genéricas de técnicas:

- *Recuperación local.* Los protocolos de recuperación local permiten recuperarse de los errores de bit en el lugar y en el momento en que esos errores se producen (por ejemplo,

en el enlace inalámbrico); un caso sería el protocolo ARQ 802.11 que hemos estudiado en la Sección 6.3, o algunas técnicas más sofisticadas que utilizan tanto ARQ como FEC [Ayanoglu 1995].

- *Conocimiento por parte del emisor TCP de enlaces inalámbricos.* En las técnicas de recuperación local, el emisor TCP es afortunadamente inconsciente de que sus segmentos están atravesando un enlace inalámbrico. Una técnica alternativa es que es el emisor y el receptor TCP sean conscientes de la existencia de un enlace inalámbrico, para distinguir entre las pérdidas por congestión que tienen lugar en la red cableada y las pérdidas/corrupciones que se producen en el enlace inalámbrico, y para invocar los mecanismos de control de congestión sólo en respuesta a las pérdidas debidas a que la red cableada está congestionada. [Balakrishnan 1997] investiga diversos tipos de TCP, bajo la suposición de que los sistemas terminales puedan llevar a cabo esta distinción. [Wei 2004] investiga técnicas para distinguir entre las pérdidas de los segmentos cableados e inalámbricos de una ruta terminal a terminal.
- *Técnicas de conexión dividida.* En la técnica de conexión dividida [Bakre 1995], la conexión terminal a terminal entre el usuario móvil y el otro punto terminal se divide en dos conexiones de la capa de transporte: una desde el host móvil hasta el punto de acceso inalámbrico y otra desde el punto de acceso inalámbrico hasta el otro punto terminal de la comunicación (que asumiremos aquí que se trata de un host cableado). La conexión terminal a terminal se forma, por tanto, mediante la concatenación de una parte inalámbrica y una parte cableada. La capa de transporte a través del segmento inalámbrico puede ser una conexión TCP estándar [Bakre 1995] o un protocolo de recuperación de errores especialmente adaptado ejecutándose sobre UDP. [Yavatkar 1994] investiga el uso de un protocolo de repetición selectiva de la capa de transporte a través de la conexión inalámbrica. Las medidas contenidas en [Wei 2006] indican que las conexiones TCP divididas se utilizan ampliamente en las redes de datos celulares y que pueden, de hecho, llevarse a cabo mejoras significativas utilizando ese tipo de conexiones divididas.

Nuestro tratamiento de TCP sobre los enlaces inalámbricos ha sido necesariamente breve. Animamos al lector a consultar las referencias para conocer más detalles acerca de esta área activa de investigación.

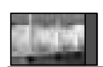
Habiendo considerado los protocolos de la capa de transporte, veamos a continuación el efecto de la tecnología inalámbrica y de la movilidad sobre los protocolos de la capa de aplicación. Aquí, una consideración importante es que los enlaces inalámbricos suelen tener anchos de banda relativamente bajos, como vimos en la Figura 6.2. Como resultado, las aplicaciones que operan sobre enlaces inalámbricos, particularmente, sobre enlaces inalámbricos celulares, deben tratar el ancho de banda como un recurso escaso. Por ejemplo, un servidor web que esté enviando contenido a un navegador web que se ejecuta sobre un teléfono 3G probablemente no pueda proporcionar un contenido con la misma riqueza de imágenes que el que envía a un navegador que esté operando a través de una conexión cableada. Aunque los enlaces inalámbricos presentan desafíos en la capa de aplicación, la movilidad que permiten también hace posible un amplio conjunto de aplicaciones conscientes de la ubicación y del contexto [Chen 2000]. En términos más generales, las redes inalámbricas y móviles desempeñarán un papel crucial a la hora de conseguir disponer de los entornos de computación ubicua del futuro [Weiser 1991]. Dicho esto, terminemos dejando claro que esto es sólo la punta del iceberg en lo que respecta al impacto de las redes inalámbricas y móviles sobre las aplicaciones en red y sus protocolos.

6.9 Resumen

Las redes inalámbricas y móviles han revolucionado la telefonía y están teniendo un impacto cada vez más profundo también en el mundo de las redes de computadoras. Con su acceso no restringido, en cualquier momento y en cualquier lugar, a la infraestructura global de red, no sólo están haciendo que el acceso de red sea más ubicuo, sino que también permiten un conjunto muy excitante de nuevos servicios dependientes de la ubicación. Dada la creciente importancia de las redes inalámbricas y móviles, este capítulo se ha centrado en los principios, en las tecnologías de enlaces comunes y en las arquitecturas de red necesarias para dar soporte a las comunicaciones inalámbricas y móviles.

Hemos comenzado el capítulo con una introducción a las redes inalámbricas y móviles, estableciendo una importante distinción entre los desafíos planteados por la naturaleza *inalámbrica* de los enlaces de comunicaciones de tales redes y por la *movilidad* que estos enlaces inalámbricos hacen posible. Esto nos ha permitido aislar, identificar y dominar mejor los conceptos clave de cada área. Nos hemos centrado primero en la comunicación inalámbrica, considerando las características de un enlace inalámbrico en la Sección 6.2. En las Secciones 6.3 y 6.4 hemos examinado los aspectos del nivel de enlace del estándar para redes LAN inalámbricas 802.11 del IEEE (WiFi), del estándar WiMAX 802.16, del estándar Bluetooth 802.15.1 y del acceso celular a Internet. Después hemos vuelto nuestra atención hacia el problema de la movilidad. En la Sección 6.5 hemos identificado diversas formas de movilidad, viendo que los diversos puntos a lo largo de este espectro planteaban diferentes desafíos y admitían distintas soluciones. Hemos considerado los problemas de localizar un usuario móvil y efectuar el enrutamiento hasta él, así como las técnicas para transferir a los usuarios móviles que se desplazan dinámicamente desde un punto de asociación con la red hasta otro. En las Secciones 6.6 y 6.7, respectivamente, hemos examinado cómo se resolvían estas cuestiones en el estándar IP móvil y en GSM. Por último, hemos considerado el impacto de los enlaces inalámbricos y la movilidad en los protocolos de la capa de transporte y sobre las aplicaciones en red en la Sección 6.8.

Aunque hemos dedicado un capítulo completo al estudio de las redes inalámbricas y móviles, haría falta un libro completo (o más) para explorar totalmente este campo tan atractivo y en tan rápida expansión. Animamos al lector a profundizar más en este campo consultando las muchas referencias que hemos proporcionado a lo largo del capítulo.



Problemas y cuestiones de repaso

Capítulo 6 • Cuestiones de repaso

SECCIÓN 6.1

R1. ¿Qué quiere decir que una red inalámbrica está operando en "modo de infraestructura"? Si la red no se encuentra en modo de infraestructura, ¿en qué modo de operación se encuentra y cuál es la diferencia entre ese modo de operación y el modo de infraestructura?

R2. ¿Cuáles son los cuatro tipos de redes inalámbricas identificados en nuestra taxonomía de la Sección 6.1? ¿Cuáles de estos tipos de redes inalámbricas ha usado usted?

SECCIÓN 6.2

R3. ¿Cuáles son las diferencias entre los siguientes tipos de deficiencias de los canales inalámbricos: pérdida de la ruta, propagación multicamino, interferencia de otros orígenes?

R4. A medida que un nodo móvil se va alejando de una estación base, ¿qué dos acciones puede llevar a cabo una estación base para garantizar que la probabilidad de pérdida de las tramas transmitidas no se incremente?

SECCIÓN 6.3

R5. Describa el papel de las tramas baliza en 802.11.

R6. Verdadero o falso: antes de que una estación 802.11 transmita una trama de datos, debe en primer lugar enviar una trama RTS y recibir la correspondiente trama CTS.

R7. ¿Por qué se utilizan los reconocimientos en 802.11, pero no en una Ethernet cableada?

R8. Verdadero o falso: Ethernet y 802.11 utilizan la misma estructura de trama.

R9. Describa cómo funciona el umbral RTS.

R10. Suponga que las tramas RTS y CTS de IEEE 802.11 fueran tan largas como las tramas DATA y ACK estándar. ¿Proporcionaría entonces alguna ventaja el uso de la tramas CTS y RTS? ¿Por qué?

R11. En la Sección 6.3.4 se ha analizado la movilidad en los estándares 802.11, en la que una estación inalámbrica se desplaza desde una BSS a otra dentro de una misma subred. Cuando los puntos de acceso están interconectados mediante un conmutador, un punto de acceso puede necesitar enviar una trama con una dirección MAC suplantada para hacer que el conmutador reenvíe la trama adecuadamente. ¿Por qué?

R12. ¿Cuáles son las diferencias entre un dispositivo maestro en una red Bluetooth y una estación base en una red 802.11?

R13. Verdadero o falso: en WiMAX, una estación base debe transmitir hacia todos los nodos a la misma velocidad de canal.

R14. ¿Qué quiere decir el término "planificación oportunista" en WiMAX?

R15. En la Sección 6.3.2 hemos visto que hay dos estándares 3G principales: UMTS y CDMA-2000. ¿De qué estándares 2G y 2.5G son herederos cada uno de estos dos estándares?

SECCIÓN 6.5–6.6

R16. Si un nodo dispone de una conexión inalámbrica con Internet, ¿tiene que ser móvil ese nodo? Explique su respuesta. Suponga que un usuario con una computadora portátil va paseando por su domicilio y siempre accede a Internet a través del mismo punto de acceso. ¿Es este usuario móvil desde el punto de vista de la red? Explique su respuesta.

R17. ¿Cuál es la diferencia entre una dirección permanente y una dirección cedida (COA)? ¿Quién asigna una dirección COA?

R18. Considere una conexión TCP que pasa a través de IP móvil. Indique si es verdadera o falsa la siguiente afirmación: la fase de la conexión TCP entre el corresponsal y el host móvil pasa a través de la red propia del móvil, pero la fase de transferencia de datos se lleva a cabo directamente entre el corresponsal y el host móvil, sin pasar por la red propia.

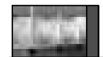
SECCIÓN 6.7

R19. ¿Cuáles son los objetivos de los registros HLR y VLR en las redes GSM? ¿Qué elementos de IP móvil son similares al HLR y al VLR?

R20. ¿Cuál es el papel del MSC ancla en las redes GSM?

SECCIÓN 6.8

R21. ¿Qué tres técnicas pueden utilizarse para evitar que un único enlace inalámbrico degrade el rendimiento de una conexión TCP terminal a terminal de la capa de transporte?



Problemas

P1. Considere el ejemplo de CDMA con un único emisor de la Figura 6.5. ¿Cuál sería la salida del emisor (para los 2 bits de datos mostrados) si el código CDMA del emisor fuera (1, –1, 1, –1, 1, –1, 1, –1)?

P2. Considere el emisor 2 en la Figura 6.6. ¿Cuál es la salida del emisor hacia el canal (antes de que se sume con la señal procedente del emisor 1), $Z^2_{i,m}$?

P3. Suponga que el receptor de la Figura 6.6 deseara recibir los datos que están siendo enviados por el emisor 2. Demuestre (mediante los cálculos necesarios) que el receptor puede efectivamente recuperar los datos del emisor 2 a partir de la señal agregada del canal utilizando el código correspondiente al emisor 2.

P4. Para el ejemplo de dos emisores y dos receptores, proporcione un ejemplo de dos códigos CDMA que contengan valores 1 y −1 y que no permitan a los dos receptores extraer los bits originales transmitidos por los emisores CDMA.

P5. Suponga que hay dos ISP que proporcionan acceso WiFi en una determinada cafetería, operando cada uno de esos ISP con su propio punto de acceso y disponiendo de su propio bloque de direcciones IP.

a. Suponga además que, por accidente, cada ISP ha configurado su punto de acceso para operar con el canal 11. ¿Dejaría completamente de funcionar el protocolo 802.11 en esta situación? Explique lo que sucede cuando dos estaciones, cada una de ellas asociada con un ISP diferente, tratan de transmitir al mismo tiempo.

b. Suponga ahora que uno de los puntos de acceso opera a través del canal 1 y que el otro opera a través del canal 11. ¿Cómo modifica esto sus respuestas anteriores?

P6. En el paso 4 del protocolo CSMA/CA, una estación que transmite con éxito una trama inicia el protocolo CSMA/CA para transmitir una segunda trama en el paso 2 en lugar de en el paso 1. ¿Qué razones pueden haber tenido en mente los diseñadores de

CSMA/CA para hacer que dicha estación no transmita la segunda trama de forma inmediata (si se detecta que el canal está inactivo)?

P7. Suponga que configuramos una estación 802.11b para reservar siempre el canal con la secuencia RTS/CTS. Suponga también que esta estación desea de repente transmitir 1.000 bytes de datos y que todas las demás estaciones están inactivas en ese momento. Calcule el tiempo requerido para transmitir la trama y recibir el mensaje de reconocimiento en función de SIFS y DIFS, e ignorando el retardo de propagación y suponiendo que no se produce ningún error de bit.

P8. Considere el escenario mostrado en la Figura 6.33, en el que hay cuatro nodos inalámbricos, A, B, C y D. El radio de cobertura de los cuatro nodos se muestra mediante los óvalos sombreados; todos los nodos comparten la misma frecuencia. Cuando A transmite, sólo puede ser escuchada/recibida por B; cuando B transmite, tanto A como C pueden escuchar/recibir desde B; cuando C transmite, tanto B como D pueden escuchar/recibir desde C; cuando D transmite, sólo C puede escuchar/recibir desde D.

Suponga ahora que cada nodo tiene un suministro infinito de mensajes que quiere enviar a cada uno de los otros nodos. Si el destino de un mensaje no es un vecino inmediato del nodo, entonces los mensajes deben ser reenviados. Por ejemplo, si A desea enviar a D, el mensaje debe enviarse primero a B, el cual lo envía a C y éste lo envía a D. El tiempo está particionado y el tiempo de transmisión de cada mensaje es exactamente igual a una partición de tiempo, como sucede por ejemplo en el protocolo Aloha con particiones. Durante una partición de tiempo, un nodo puede hacer una de las cosas siguientes: (*i*) enviar un mensaje; (*ii*) recibir un mensaje (si se está enviando exactamente un mensaje a ese nodo), (*iii*) permanecer en silencio. Como siempre, si un nodo escucha dos o más transmisiones simultáneas se produce una colisión y ninguno de los mensajes transmitidos será recibido correctamente. Puede asumir aquí que no existen errores de nivel de bit y que, por tanto, si se está enviando exactamente un mensaje, éste será recibido correctamente por aquellos nodos que se encuentren dentro del radio de transmisión del emisor.

a. Suponga ahora que un controlador omnisciente (es decir, un controlador que conoce el estado de todos los nodos de la red) puede ordenar a cada nodo que haga lo que el controlador omnisciente quiere, es decir, enviar un mensaje, recibir un mensaje o permanecer en silencio. Dado este controlador omnisciente, ¿cuál es la velocidad máxima a la que puede transmitirse un mensaje desde C a A, supuesto que no haya ningún otro mensaje entre ningún otro par de nodos origen/destino?

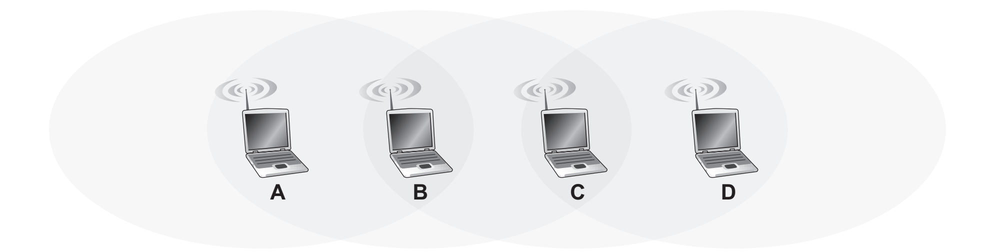


Figura 6.33 • Escenario para el Problema P8.

b. Suponga ahora que A envía mensajes a B y que D envía mensajes a C. ¿Cuál es la tasa máxima combinada a la que pueden fluir mensajes de datos desde A hasta B y desde D hasta C?

c. Suponga ahora que A envía mensajes a B y que C envía mensajes a D. ¿Cuál es la tasa máxima combinada a la que pueden fluir mensajes de datos desde A hasta B y desde C hasta D?

d. Suponga ahora que sustituimos los enlaces inalámbricos por enlaces cableados. Repita los apartados (a) hasta (c) en este escenario cableado.

e. Ahora suponga que volvemos al escenario inalámbrico y que para todos los mensajes de datos enviados desde el origen al destino, el destino responde con un mensaje ACK dirigido al origen (por ejemplo, como en TCP). Suponga también que cada mensaje ACK ocupa una partición. Repita los apartados (a) – (c) anteriores para este escenario.

P9. Describa el formato de la trama Bluetooth 802.15.1. Tendrá que leer algunas referencias, además de este libro, para poder encontrar la información correspondiente. ¿Hay algo en el formato de trama que limite de manera inherente el número de nodos activos de una red 802.15.1 a ocho nodos? Explique su respuesta.

P10. Considere el siguiente escenario WiMAX idealizado. La subtrama descendente (véase la Figura 6.17) está particionada en el tiempo, con *N* particiones descendentes por subtrama, teniendo todas las particiones de tiempo la misma duración. Hay cuatro nodos, A, B, C y D, alcanzables desde la estación base a velocidades de 10 Mbps, 5 Mbps, 2,5 Mbps y 1 Mbps, respectivamente, en el canal descendente. La estación base tiene una cantidad infinita de datos que enviar a cada uno de los nodos y puede enviar a cualquiera de esos cuatro nodos durante cualquiera de las particiones de tiempo de la subtrama descendente.

a. ¿Cuál es la velocidad máxima a la que la estación base puede enviar a los nodos, suponiendo que puede enviar a cualquier nodo que elija durante cada partición de tiempo? ¿Es equitativa su solución? Explique su respuesta y defina qué quiere decir con el término "equitativa".

b. Si imponemos el requisito de que la comunicación sea equitativa, en el sentido de que cada nodo deba recibir la misma cantidad de datos durante cada subtrama descendente, ¿cuál es la velocidad media de transmisión de la estación de base (a todos los nodos) durante la subtrama descendente? Explique cómo ha llegado a obtener su respuesta.

c. Suponga que el criterio de comunicación equitativa es que cada nodo puede recibir como mucho el doble de datos que cualquier otro nodo durante la subtrama. ¿Cuál es la velocidad media de transmisión de la estación base (a todos los nodos) durante la subtrama? Explique cómo ha obtenido su respuesta.

P11. En la Sección 6.5, una solución propuesta para permitir que los usuarios móviles mantengan sus direcciones IP a medida que se desplazan entre redes ajenas era hacer que una red ajena anunciara una ruta altamente específica hacia el usuario móvil y usar la infraestructura de enrutamiento existente para propagar esta información a través de la red. Para esta solución, ya dijimos que la escalabilidad es un problema. Suponga que cuando un usuario móvil se desplaza de una red a otra, la nueva red ajena anuncia una

ruta específica hacia el usuario móvil y que la red ajena antigua retira su ruta. Considere cómo se propagaría la información de enrutamiento con un algoritmo de vector de distancia (particularmente para el caso de enrutamiento entre dominios, entre redes que abarquen todo el globo terráqueo).

a. ¿Podrían otros routers ser capaces de enrutar datagramas inmediatamente hacia la nueva red ajena, en cuanto la red ajena comience a anunciar su ruta?
b. ¿Es posible que los diferentes routers crean que el usuario móvil se encuentra en diferentes redes ajenas?
c. Analice la escala temporal sobre la que otros routers de la red terminarán por aprender la ruta hacia los usuarios móviles.

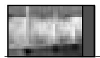
P12. Suponga que el corresponsal de la Figura 6.22 fuera móvil. Dibuje la infraestructura adicional de la capa de red que sería necesaria para enrutar el datagrama desde el usuario móvil original hasta el (ahora móvil) corresponsal. Muestre la estructura del datagrama (o de los datagramas) entre el usuario móvil original y el (ahora móvil) corresponsal, como en la Figura 6.23.

P13. En IP móvil, ¿qué efecto tendrá la movilidad sobre los retardos terminal a terminal de los datagramas enviados desde un origen a un destino?

P14. Considere el ejemplo de encadenamiento analizado al final de la Sección 6.7.2. Suponga que un usuario móvil visita las redes ajenas A, B y C y que un corresponsal inicia una conexión con el usuario móvil cuando éste reside en la red ajena A. Enumere la secuencia de mensajes entre los agentes ajenos y entre esos agentes ajenos y el agente propio, a medida que el usuario móvil se desplaza desde la red A a la red B y a la red C. A continuación, suponga que no realizamos el encadenamiento y que hay que notificar explícitamente al corresponsal (así como al agente propio) los cambios en la dirección COA del usuario móvil. Enumere la secuencia de mensajes que habría que intercambiar en este segundo escenario.

P15. Considere dos nodos móviles en una red ajena que dispone de un agente ajeno. ¿Es posible que los dos nodos móviles utilicen la misma dirección COA en IP móvil? Explique su respuesta.

P16. En nuestro análisis de cómo el registro VLR actualizaba el HLR con información acerca de la ubicación actual del móvil, ¿cuáles son las ventajas y desventajas de proporcionar al HLR el MSRN en lugar de la dirección del VLR?



Preguntas para la discusión

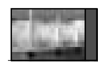
D1. Enumere cinco productos que estén actualmente en el mercado y que proporcionen una interfaz Bluetooth 802.15.

D2. ¿Hay servicio inalámbrico 3G disponible en su región? ¿Qué precio tiene? ¿Qué aplicaciones soporta?

D3. Como usuario de IEEE 802.11, ¿qué tipos de problemas ha observado? ¿Cómo pueden evolucionar los diseños de 802.11 para solventar estos problemas?

D4. Realice una búsqueda en la Web para localizar pruebas de implantación de WiMAX. ¿Qué extensión han tenido estos proyectos piloto? ¿Qué velocidades se han conseguido y a qué distancias? ¿Con cuántos usuarios?

D5. Realice una búsqueda en la Web para ver implantaciones de EVDO y HSDPA. ¿Cuál ha sido la tecnología más implantada hasta la fecha? ¿Dónde?

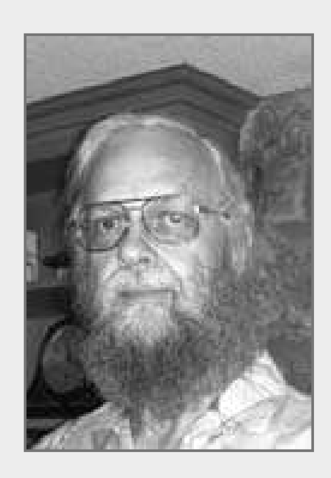


Prácticas de laboratorio con Wireshark

En el sitio web del libro, http://www.awl.com/kurose-ross, encontrará una práctica de laboratorio con Wireshark para este capítulo en la que se capturan y analizan las tramas 802.11 intercambiadas entre una computadora portátil inalámbrica y un punto de acceso.

UNA ENTREVISTA CON...

Charlie Perkins



Charles E. Perkins es Technical Fellow en WiChorus, donde se dedica a investigar nuevas técnicas para la aplicación de protocolos de gestión de la movilidad en Internet a nuevas generaciones de medios inalámbricos, como WiMAX y LTE. Estas tecnologías ya han comenzado a hacer que las conexiones inalámbricas de banda ancha sean una realidad, abriendo nuevas oportunidades para la expansión de Internet y para el suministro de contenido multimedia a la carta. Es el editor documental del grupo de trabajo de IP móvil del IETF (*Internet Engineering Task Force*), autor o co-autor de documentos de estandarización en los grupos de trabajo mip4, mip6, manet, mext, dhc, seamoby (movilidad transparente) y autoconf, y es editor de diversas revistas de la ACM y el IEEE relacionadas con las redes inalámbricas. En WiChorus, ha continuado su implicación en desarrollos avanzados que emplean IP móvil, IPv6 y otros protocolos basados en los trabajos del IETF. Ha sido autor y editor de libros sobre IP móvil y redes ad hoc, y ha publicado diversos artículos, algunos de ellos premiados, en las áreas de redes móviles, redes ad hoc, optimización de rutas para redes móviles, descubrimiento de recursos y configuración automática para computadoras móviles. Es uno de los creadores de MobiHoc y ha actuado como General Chair y Program Committee Chair. Ha participado en el Consejo de Arquitectura Internet del IETF y en varios comités para el Consejo Nacional de Investigación de los Estados Unidos, y también ha formado parte de diversos paneles de evaluación técnica para el Laboratorio de Investigación del Ejército y el programa MICS suizo. Más recientemente, ha actuado como General Chair en la conferencia WiNS-DR y de MASS 2006.

¿Por qué decidió especializarse en tecnología inalámbrica/móvil?

Mi implicación con las redes inalámbricas y la movilidad fue una consecuencia natural de mi trabajo en proyectos del departamento de investigación de IBM a finales de la década de 1980. Disponíamos de enlaces vía radio y estábamos intentando construir un tipo de dispositivo al estilo del “ThinkPad” (como Palm Pilot) con conectividad inalámbrica y reconocimiento de escritura manuscrita.

Desarrollamos una solución simple (posteriormente denominada IP móvil) y nos dimos cuenta de que funcionaba. Utilizando nuestra experiencia con IP móvil, desarrollamos una modificación rápida y efectiva de RIP que permitía construir redes ad hoc. Esto también funcionó muy bien. Al decir que “funcionó”, quiero decir que las aplicaciones funcionaban perfectamente sin ninguna modificación y que la red no se venía abajo a causa de los nuevos diseños. Estas propiedades son las que se suelen denominar “transparencia de aplicación” y “escalabilidad”.

Por supuesto, el trabajo en un laboratorio es enormemente distinto del éxito comercial, y estas dos tecnologías tienen todavía un gran potencial comercial que no se ha visto satisfecho.

¿Cuál fue su primer trabajo en la industria de las computadoras?

Trabajé en TRW Controls, en Houston, Texas. Fue un cambio bastante drástico con respecto a los estudios universitarios.

Una cosa que aprendí en TRW Controls es lo pobre que es el software de soporte incluso para los sistemas de control de utilidades más críticos. Se supone que estos sistemas debían controlar el flujo de electricidad en enormes redes eléctricas, y el software subyacente era desarrollado de una manera que ponía los pelos de punta. Además, los planes de desarrollo tenían siempre unos tiempos muy apretados y los programadores eran enormemente cínicos acerca de las intenciones de la dirección y de las condiciones de trabajo.

El sistema completo necesitaba ser rediseñado desde el principio. No tengo muchas razones para creer que las cosas hayan cambiado durante los últimos 30 años, especialmente dadas la recientes noticias acerca del apagón de 2003. De hecho, dada la desregulación que ha vivido el sector, estoy seguro de que las cosas son todavía peor de como las recuerdo.

Me satisfizo mucho dejar TRW Controls y entrar a trabajar en Tektronix (Tek Labs).

¿Cuál es la parte de su trabajo que constituye un mayor desafío?

El mayor reto de mi trabajo es comprender qué es lo que debo hacer para ayudar a mi empresa. Asimismo, considero parte de mi trabajo dar forma a las tecnologías inalámbricas con las que entro en contacto, para que puedan proporcionar un mejor servicio y una experiencia cotidiana más agradable a la gente. El negocio de mi empresa consiste en proporcionar equipos de infraestructura para la conectividad inalámbrica de alta velocidad. Además de hacer evolucionar los documentos de estandarización relevantes, espero encontrar formas de simplificar los sistemas resultantes aplicando las diversas técnicas relacionadas con los trabajos del IETF. Hacer esto de una forma que permita también maximizar el beneficio potencial de las tecnologías que desarrollamos convierte cada día en un nuevo desafío. Son muchas las cosas que hay que hacer y las oportunidades son inmensas.

A un nivel técnico más detallado, en el que por cierto me encuentro mucho más cómodo, trato de resolver problemas de los protocolos de red de forma que impongan la menor carga posible a los dispositivos inalámbricos (¡y a sus baterías!) y que presenten a los usuarios las mínimas incomodidades posibles. Interconectar los dispositivos inalámbricos actuales con Internet mediante las nuevas tecnologías inalámbricas de alta velocidad es enormemente interesante desde el punto de vista técnico y ofrece un potencial ilimitado para el éxito comercial para aquéllos que puedan encontrar la manera correcta de proseguir los desarrollos. Además, ahora estamos comenzando a enfrentarnos a un interesante desafío, a medida que nuestro espacio de direcciones IPv4 subyacente se empieza a agotar, previéndose que ese agotamiento tendrá lugar en los próximos dos o tres años. IPv6 ha resultado bastante más difícil de implantar de lo que se predijo hace diez años, aún cuando las especificaciones básicas estén bastante maduras.

¿Cuál cree que es el futuro de la tecnología inalámbrica?

Toda la industria inalámbrica está sufriendo un tremendo cambio y nadie sabe cuál puede ser el fin. Están emergiendo tecnologías inalámbricas de alta velocidad y pueden tener efectos prácticos no previstos que podrían cambiar de forma importante nuestra sociedad. Nuestras actuales expectativas acerca de la intimidad y las limitaciones en nuestra capacidad de comunicarnos unos con otros (voz, imágenes y datos), podrían ser irreconocibles dentro de diez años. A medida que las empresas se vayan convirtiendo más y más a las comunicaciones inalámbricas, resulta bastante posible que se tomen nuevas medidas de seguridad que cambiarán significativamente nuestra experiencia de trabajo.

Parece bastante claro que se asignarán más partes del espectro a los distintos esquemas de las comunicaciones por radio. Los nuevos esquemas pueden tener una muy alta velocidad. Hay comunidades que han hecho experimentos de ofrecer a sus ciudadanos más y más comunicaciones inalámbricas de alta velocidad; una ciudad completa podría convertirse en una red inalámbrica de área local.

Esto podría tener el efecto de reforzar el sentimiento de comunidad que se ha perdido hace mucho en nuestra sociedad, al menos en Estados Unidos. Por supuesto, la comunidad seguirá demandando acceso a Internet. La capacidad de disco está creciendo tan rápidamente y a precios tan económicos, que ya podemos llevar en nuestros bolsillos toda la Wikipedia y probablemente todos los número de teléfono del mundo, por no mencionar bibliotecas personales sin precedentes llenas de libros, música y películas.

La tecnología inalámbrica está acelerando el crecimiento de Internet. A medida que se abaratan los dispositivos inalámbricos, estamos viendo comunicaciones Internet por todas partes (pendientes, juegos multijugador, lectores de billetes en el metro). Esto está motivando la aparición de nuevas aplicaciones y nuevas soluciones de seguridad.



CAPÍTULO 7

Redes multimedia

Actualmente estamos siendo testigos de la amplia implantación de aplicaciones de audio y vídeo en Internet. Cientos de sitios, entre los que se incluyen CCN, Rhapsody, Napster, MSN, AOL, Yahoo!, ponen a nuestra disposición flujos con contenido de audio y vídeo. YouTube y otros sitios de compartición de vídeo permiten a los usuarios utilizar (bajo demanda) clips de vídeo que han sido cargados por otros usuarios. Millones de usuarios utilizan de forma regular Skype para cubrir sus necesidades de telefonía y de videoconferencia. Y algunos canales de televisión tradicionales ahora están distribuyendo a través de Internet, permitiendo a los usuarios de Internet ver canales de televisión que tienen su origen en todos los rincones del mundo. Este explosivo crecimiento de las aplicaciones multimedia en Internet es principalmente el resultado de la creciente penetración del acceso residencial a la banda ancha y el acceso inalámbrico de alta velocidad (como por ejemplo WiFi). Como hemos visto en la Sección 1.2, las velocidades de acceso de banda ancha continuarán aumentando, animando aún más la implantación de nuevas y excitantes aplicaciones multimedia.

Los requisitos de servicio de las aplicaciones multimedia difieren significativamente de los de las aplicaciones elásticas tradicionales, como el correo electrónico, la navegación por la Web, los inicios de sesión remotos y la descarga y compartición de archivos (que hemos estudiado en el Capítulo 2). En particular, a diferencia de las aplicaciones elásticas, las aplicaciones multimedia son extremadamente sensibles a los retardos terminal a terminal y a la variación del retardo, aunque pueden tolerar pérdidas de datos ocasionales.

Comenzaremos este capítulo con una taxonomía de las aplicaciones multimedia en la Sección 7.1. Veremos que una aplicación multimedia puede clasificarse como *flujo de audio/vídeo almacenado, flujo de audio/vídeo en vivo* o *audio/vídeo interactivo en tiempo real*. Además, veremos que cada una de estas clases de aplicación tiene un conjunto diferente de requisitos de servicio para la red. En la Sección 7.2, examinaremos los flujos de audio/vídeo almacenado. En la Sección 7.3, investigaremos las técnicas de nivel de aplicación que pueden mejorar el rendimiento de las aplicaciones multimedia dentro del servicio de mejor esfuerzo de la red Internet actual y, en la Sección 7.4, nos ocuparemos de varios protocolos multimedia que se emplean actualmente en Internet. En la Sección 7.5, investi-

garemos mecanismos de la red que pueden utilizarse para diferenciar una clase de tráfico (por ejemplo, aplicaciones tolerantes al retardo tales como las multimedia) de otra (por ejemplo, aplicaciones elásticas como FTP), y proporcionaremos un servicio diferenciado entre las diversas clases de tráfico. Por último, en la Sección 7.6, consideraremos el caso en el que la red debe proporcionar *garantías* de rendimiento a las aplicaciones; por ejemplo, que una llamada telefónica IP basada en paquetes obtenga el mismo rendimiento que si la llamada se hiciera a través de una red telefónica de conmutación de circuitos. Veremos que esto requiere la introducción de nuevos mecanismos y protocolos de red.

7.1 Aplicaciones de redes multimedia

En nuestra exposición del Capítulo 2 acerca de los requisitos de servicio de las aplicaciones identificamos una serie de ejes a lo largo de los cuales pueden clasificarse estos requisitos. Dos de estos ejes (consideraciones sobre temporización y tolerancia a la pérdida de datos) son particularmente importantes para las aplicaciones multimedia en red. Las consideraciones sobre temporización son importantes porque muchas aplicaciones multimedia son extremadamente **sensibles a los retardos**. Veremos enseguida que en muchas aplicaciones multimedia los paquetes que sufren un retardo emisor-receptor de más de unos pocos cientos de milisegundos resultan inútiles para el receptor. Por otro lado, las aplicaciones multimedia de red casi siempre son **tolerantes a las pérdidas** (las pérdidas ocasionales sólo causan fallos ocasionales en la reproducción del audio/vídeo y estas pérdidas a menudo pueden ser parcial o totalmente disimuladas). Estas características de sensibilidad a los retardos y tolerancia a las pérdidas son claramente diferentes de las correspondientes relativas a las aplicaciones elásticas como la Web, el correo electrónico, FTP y Telnet. En las aplicaciones elásticas, los retardos largos son molestos aunque no especialmente dañinos, pero la completitud y la integridad de los datos transferidos son de suma importancia.

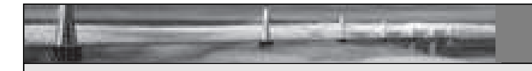
7.1.1 Ejemplos de aplicaciones multimedia

Internet puede soportar una amplia variedad de atractivas aplicaciones multimedia. En esta subsección vamos a considerar tres clases generales de aplicaciones multimedia: los flujos de audio/vídeo almacenado, los flujos de audio/vídeo en vivo y el audio/vídeo interactivo en tiempo real.

En este capítulo *no* vamos a ocuparnos de las aplicaciones que se descargan y luego se reproducen, como por ejemplo la descarga completa de un archivo MP3 mediante una aplicación de compartición de archivos P2P antes de reproducirlo. De hecho, las aplicaciones que se descargan y luego se reproducen son aplicaciones elásticas de transferencia de archivos sin ningún requisito especial de retardo. En el Capítulo 2 ya hemos examinado la transferencia de archivos (HTTP y FTP) y los sistemas de compartición de archivos P2P.

Flujos de audio y vídeo almacenado

En esta clase de aplicaciones, los clientes solicitan archivos de audio o de vídeo comprimidos bajo demanda que están almacenados en servidores. Hoy día existen miles de sitios que proporcionan flujos de audio y de vídeo almacenado, entre los que se incluyen CNN, Microsoft Video y YouTube. Este tipo de aplicaciones presentan tres características diferenciadoras:



HISTORIA

IPTV

Tradicionalmente, el contenido de televisión ha sido distribuido mediante microondas terrestres, sistemas híbridos de fibra y cable coaxial (HFC) y canales de satélites geoestacionarios (véase la Sección 1.2). Pero en la época actual de Internet existe un tremendo interés en la IPTV; es decir, la distribución del contenido de televisión a través de Internet.

Uno de los desafíos de la tecnología IPTV es la gran cantidad de ancho de banda necesario, especialmente en el servidor de origen. Por ejemplo, considere la distribución de un evento deportivo importante, como el partido de la Copa del Mundo, desde un único servidor a 100 millones de usuarios concurrentes a través de Internet. Si la velocidad de vídeo es sólo de 1 Mbps, entonces el ancho de banda requerido del servidor sería del orden de 100 terabits/segundo. Por tanto, la clásica distribución cliente-servidor es completamente inaceptable. Si la multidifusión IP estuviera ampliamente implantada en Internet sería mucho más fácil hacer de IPTV una realidad. Otra alternativa es distribuir el vídeo a través de una red solapada de multidifusión, tal como las proporcionadas por las redes de distribución de contenido (CDN, *Content Distribution Network*) (véase la Sección 7.3).

Otra alternativa más sería emplear la distribución entre pares, donde cada par que recibe un canal de televisión también ayuda a redistribuir el canal a otros pares. Quizá el mayor atractivo de esta técnica es el bajo coste de distribución: si los pares individuales proporcionan colectivamente el suficiente ancho de banda de carga, es posible que se necesite poco ancho de banda en el servidor (quizá sólo unas cuantas veces la velocidad de vídeo). Con un coste tan bajo, cualquier usuario equipado con una cámara web podría distribuir un programa en vivo a millones de usuarios a un coste despreciable.

Hasta la fecha, una serie de sistemas de IPTV P2P similares a BitTorrent han disfrutado de una exitosa implantación. El pionero en este campo, CoolStreaming, tenía más de 4.000 usuarios simultáneos en 2003 [CoolStreaming 2005]. Más recientemente, otros sistemas, entre los que se incluyen PPLive y ppstream, han informado de grandes éxitos, con decenas de miles de usuarios simultáneos viendo canales a velocidades de entre 300 kbps y 1 Mbps. En estos sistemas de tipo BitTorrent, los pares forman una red solapada dinámica e intercambian fragmentos de vídeo con sus vecinos de red. Será interesante ver cómo evoluciona IPTV en los próximos 5 a10 años. ¿Qué tecnología subyacente utilizará: CDN o P2P, o alguna que sea un híbrido de ambas? ¿Habrá una parte significativa de los aficionados a la Copa del Mundo que vean los partidos a través de Internet en 2014?

- *Medios almacenados.* El contenido multimedia, que ha sido pregrabado, se almacena en un servidor. Puesto que el contenido está pregrabado, el usuario en el cliente puede pausar, rebobinar, pasar hacia adelante o buscar en el contenido multimedia. El instante desde que el usuario hace una de estas solicitudes hasta que la acción tiene lugar en el cliente debe ser del orden de uno a diez segundos para ser una respuesta aceptable.
- *Flujos.* En una aplicación de flujo de audio/vídeo almacenado, normalmente un cliente inicia la reproducción del audio/vídeo y unos pocos segundos después comienza a recibir el archivo procedente del servidor. Esto significa que el cliente reproducirá el audio/

vídeo desde una posición del archivo mientras está recibiendo del servidor partes posteriores del mismo. Esta técnica, conocida como **transmisión de flujos**, evita descargar el archivo completo (e incurrir en un retardo potencialmente largo) antes de comenzar la reproducción. Existen muchos clientes para la reproducción de flujos multimedia como RealPlayer de RealNetworks [RealNetworks 2009], QuickTime de Apple [QuickTime 2009], y Windows Media de Microsoft [Microsoft Media Player 2009].

- *Reproducción continua*. Una vez que se inicia la reproducción del contenido multimedia debería proseguir de acuerdo con la temporización original de la grabación. Por tanto, los datos deben recibirse del servidor a tiempo para su reproducción en el cliente; si no es así, el usuario experimentará frustrantes retardos de almacenamiento en buffer. Aunque las aplicaciones multimedia almacenadas tienen que cumplir los requisitos de reproducción continua, sus restricciones de retardo terminal a terminal son menos restrictivas que las de las aplicaciones interactivas en vivo, como la telefonía y la videoconferencia por Internet (véase más adelante).

Flujos de audio y vídeo en vivo

Este tipo de aplicaciones es similar a la difusión tradicional de radio y televisión, excepto porque la transmisión tiene lugar a través de Internet. Estas aplicaciones permiten a un usuario recibir una transmisión de radio o televisión *en vivo* emitida desde cualquier rincón del mundo. (Por ejemplo, uno de los autores de este libro con frecuencia escucha su emisora de radio favorita de Filadelfia cuando viaja. El otro autor normalmente escuchaba las difusiones en vivo de los partidos de su muy querido equipo de baloncesto de la universidad cuando estuvo viviendo en Francia durante un año.) Estas aplicaciones se conocen como IPTV y radio por Internet. Hoy día, existen miles de emisoras de radio que emiten a través de Internet y un gran número de implantaciones de IPTV (véase el recuadro dedicado a IPTV).

Dado que los flujos de audio/vídeo en vivo no están almacenados, un cliente no puede hacer un avance rápido del medio. Sin embargo, con el almacenamiento local de los datos recibidos otras operaciones interactivas tales como la puesta en pausa o el rebobinado sí son posibles. Las aplicaciones en vivo de tipo difusión a menudo tienen muchos clientes que reciben el mismo programa de audio/vídeo. La distribución de audio/vídeo en vivo a muchos receptores puede llevarse a cabo de forma eficiente mediante las técnicas de multidifusión IP descritas en la Sección 4.7. Sin embargo, actualmente la distribución de audio/vídeo en vivo suele conseguirse a través de flujos de multidifusión de la capa de aplicación (utilizando P2P o CDN) o a través de múltiples flujos de unidifusión servidor-cliente separados. Al igual que con los flujos multimedia almacenados se requiere reproducción continua, aunque las restricciones de temporización son menos estrictas que para las aplicaciones interactivas en tiempo real. Pueden tolerarse retardos de hasta decenas de segundos desde el momento en que el usuario solicita el suministro/reproducción de una transmisión en vivo hasta que comienza la reproducción.

Audio y vídeo interactivo en tiempo real

Este tipo de aplicaciones permite a los usuarios emplear el audio y el vídeo para comunicarse entre sí en tiempo real. El audio interactivo en tiempo real a través de Internet suele referirse como **telefonía por Internet**, ya que, desde la perspectiva del usuario, es similar al servicio de telefonía de conmutación de circuitos tradicional. La telefonía por Internet puede proporcionar centrales telefónicas PBX (*Private Branch Exchange*) y servicios telefónicos

locales y de larga distancia a un coste muy bajo. También puede facilitar la implantación de nuevos servicios que las redes tradicionales de conmutación de circuitos no soportan fácilmente, como por ejemplo la detección de presencia, la comunicación en grupo, el filtrado de llamantes, la integración telefonía-Web y muchos más. Actualmente hay disponibles numerosos productos de tele-fonía Internet. Por ejemplo, los usuarios de Skype pueden realizar llamadas de voz PC-a-teléfono y PC-a-PC. Con el vídeo interactivo en tiempo real, también denominado videoconferencia, los individuos pueden comunicarse de forma visual y oral. Actualmente también hay disponibles muchos productos de vídeo interactivo en tiempo real para Internet, entre los que se incluyen NetMeeting de Microsoft, vídeo Skype y diversos productos de Polycom. Observe que en una aplicación de audio/vídeo interactivo en tiempo real un usuario puede hablar o mover la cabeza en cualquier momento. En una conversación con interacción entre varios interlocutores, el retardo desde que un usuario habla o se mueve hasta que la acción se manifiesta en los hosts receptores debe ser menor que unos pocos cientos de milisegundos. En el caso de voz, los retardos menores de 50 milisegundos no son percibidos por el oído humano, los retardos comprendidos entre 150 y 400 milisegundos pueden ser aceptables y los retardos mayores de 400 milisegundos pueden dar lugar a conversaciones frustrantes, si no completamente ininteligibles.

7.1.2 Obstáculos para la información multimedia en la Internet actual

Recuerde que el protocolo IP implantado en la red Internet actual proporciona un **servicio de mejor esfuerzo** a todos los datagramas que transporta. En otras palabras, Internet hace todo lo posible por trasladar cada datagrama desde el emisor al receptor tan rápido como es posible, pero no promete nada sobre el retardo terminal a terminal que puede sufrir un paquete individual. Ni tampoco hace ninguna promesa acerca de la variación del retardo de paquete dentro de un flujo de paquetes. Puesto que TCP y UDP se ejecutan sobre IP, se deduce que ninguno de estos protocolos de transporte proporciona ninguna garantía de retardo a las aplicaciones que los invocan. Debido a la falta de un esfuerzo especial por entregar los paquetes a tiempo, es un problema extremadamente desafiante desarrollar aplicaciones de red multimedia de éxito para Internet. No obstante, hasta la fecha, las aplicaciones multimedia en Internet han alcanzado un notable éxito. Por ejemplo, los flujos de audio y de vídeo almacenado con retardos de interactividad con el usuario de entre 5 y 10 segundos son actualmente habituales en Internet. Pero durante los periodos de pico de tráfico, el rendimiento puede ser insatisfactorio, especialmente cuando los enlaces intervinientes están congestionados (como los enlaces transoceánicos congestionados).

El vídeo interactivo en tiempo real y la telefonía por Internet han encontrado también un amplio uso; por ejemplo, de forma rutinaria hay más de siete millones de usuarios de Skype en línea en un instante de tiempo determinado. Las aplicaciones de vídeo y voz interactivas en tiempo real imponen estrictas restricciones sobre el retardo y la fluctuación de los paquetes. La **fluctuación de paquetes** (*packet jitter*) es la variabilidad de los retardos de los paquetes dentro del mismo flujo de paquetes. Las aplicaciones de voz y vídeo en tiempo real pueden funcionar bien cuando el ancho de banda está completamente disponible y, por tanto, el retardo y la fluctuación son mínimos. Pero la calidad puede deteriorarse hasta unos niveles inaceptables tan pronto como el flujo de paquetes de voz o de vídeo en tiempo real llegan a un enlace moderadamente congestionado.

El diseño de aplicaciones multimedia podría ser realmente más sencillo si existieran servicios Internet de primera y segunda clase, en los que los paquetes de primera clase estu-

vieran limitados en número y recibieran un servicio de prioridad en las colas de los routers. Una servicio de primera clase de este tipo podría ser satisfactorio para las aplicaciones sensibles al retardo. Pero, hasta la fecha, Internet emplea casi siempre un método igualitario para la planificación de paquetes en las colas de los routers. Todos los paquetes reciben el mismo servicio; ningún paquete, incluyendo los paquetes de audio y vídeo sensibles al retardo, reciben un servicio especial de prioridad en las colas de los routers. No importa el dinero que se tenga ni lo importante que se sea, ¡tendrá que ponerse a la cola y esperar a que le llegue su turno! En la segunda mitad del capítulo examinaremos las arquitecturas propuestas que plantean la eliminación de esta restricción.

Así, por el momento tenemos que seguir conviviendo con el servicio de mejor esfuerzo. Pero conocida esta restricción, podemos tomar algunas decisiones de diseño y emplear algunos trucos para mejorar la calidad percibida por el usuario de aplicaciones multimedia de red. Por ejemplo, podemos enviar los datos de audio o de vídeo a través de UDP, e ignorar la baja tasa de transferencia de TCP cuando este protocolo entra en su fase de arranque lento. Podemos retardar la reproducción en el receptor 100 o más milisegundos, con el fin de atenuar los efectos de la fluctuación inducida por la red. Podemos incluir marcas de tiempo en los paquetes en el emisor, de manera que el receptor sepa cuándo deberían reproducirse los paquetes. En el caso del audio y el vídeo almacenado podemos extraer datos anticipadamente durante la reproducción siempre que existan espacios de almacenamiento en el cliente y ancho de banda adicionales. Podemos incluso enviar información redundante para mitigar los efectos de la pérdida de paquetes inducida por la red. Estudiaremos muchas de estas técnicas en la primera parte del capítulo.

7.1.3 ¿Cómo debería evolucionar Internet para dar un mejor soporte a las aplicaciones multimedia?

Hoy día existe un debate continuo acerca de cómo debería evolucionar Internet para acomodar mejor el tráfico multimedia con sus estrictas restricciones de temporización. En un extremo se encuentran algunos investigadores que argumentan que habría que llevar a cabo cambios fundamentales en Internet, con el fin de que dichas aplicaciones puedan reservar explícitamente ancho de banda terminal a terminal y recibir así una *garantía* para su rendimiento terminal a terminal. Una **garantía estricta** quiere decir que la aplicación recibirá la calidad de servicio (QoS) solicitada con absoluta seguridad. Una **garantía parcial** quiere decir que la aplicación recibirá su calidad de servicio solicitada con una alta probabilidad. Estos investigadores creen que si un usuario desea, por ejemplo, hacer una llamada telefónica por Internet desde el host A al host B, entonces la aplicación telefónica de Internet debería poder reservar ancho de banda de forma explícita en cada uno de los enlaces de una ruta existente entre ambos hosts. Pero permitir que las aplicaciones hagan reservas y requerir a la red aceptar las reservas requiere algunos grandes cambios. En primer lugar, necesitamos un protocolo que, en nombre de las aplicaciones, reserve ancho de banda en los enlaces que definen el camino desde los emisores hasta sus respectivos receptores. En segundo lugar, habrá que modificar las políticas de planificación en las colas de los routers, de modo que las reservas de ancho de banda puedan ser aceptadas. Con estas nuevas políticas de planificación, no todos los paquetes recibirán el mismo tratamiento; en su lugar, aquellos que reserven (y paguen) más obtendrán más. En tercer lugar, para aceptar las reservas, las aplicaciones tienen que proporcionar a la red una descripción del tráfico que pretenden enviar. La red entonces tendrá que vigilar el tráfico de cada aplicación para garantizar que cumple

con la descripción. Por último, la red debe disponer de un medio para determinar si tiene el suficiente ancho de banda disponible como para dar soporte a cualquier nueva solicitud de reserva. Estos mecanismos, cuado se combinan, requieren un software nuevo y complejo tanto en los hosts como en los routers, así como nuevos tipos de servicios. Veremos estos mecanismos en detalle en la Sección 7.6.

En el otro extremo, algunos investigadores argumentan que no es necesario realizar ningún cambio fundamental en el servicio de mejor esfuerzo ni en los protocolos de Internet subyacentes. En su lugar, defienden el método de *laissez-faire*:

- A medida que la demanda aumenta, los ISP (tanto de nivel superior como de nivel inferior) escalan sus redes para satisfacer la demanda. Específicamente, los ISP ofrecerán el suficiente ancho de banda y la suficiente capacidad de conmutación como para proporcionar, dentro de sus redes, un funcionamiento satisfactorio en lo que respecta al retardo y a la pérdida de paquetes [Huang 2005]. Los ISP proporcionarán un mejor servicio a sus clientes (usuarios e ISP clientes), que se traducirá en unos mayores ingresos, gracias a la existencia de más clientes y a las mayores tasas por los servicios. Para garantizar que las aplicaciones multimedia recibirán un servicio adecuado, incluso en el caso de que exista sobrecarga, un ISP puede sobredimensionar el ancho de banda y la capacidad de conmutación. Con una previsión del tráfico y una provisión de ancho de banda apropiadas puede proporcionarse una garantía parcial de calidad de servicio (QoS).
- La redes de distribución de contenido (CDN) duplican el contenido almacenado y lo insertan en las fronteras de Internet. Puesto que una gran parte del tráfico que fluye a través de Internet es contenido almacenado (páginas web, archivos MP3, vídeo), las redes CDN pueden aliviar de forma significativa las cargas de tráfico en los ISP y las interfaces entre pares entre los ISP. Además, las redes CDN proporcionan un servicio diferenciado a los proveedores de contenido: los proveedores de contenido que pagan un servicio CDN pueden suministrar el contenido más rápidamente y de forma más efectiva. En la Sección 7.3 estudiaremos las redes CDN.
- En el caso del tráfico de flujos en vivo (como el generado por un evento deportivo) que se envía a millones de usuarios simultáneamente, pueden implantarse **redes solapadas de multidifusión**. Una red de este tipo consta de hosts de usuario y, posiblemente, de servidores dedicados dispersos por Internet. Estos hosts, servidores y enlaces lógicos entre ellos forman colectivamente una red solapada, la cual multidifunde (véase la Sección 4.7) el tráfico desde el emisor a millones de usuarios. A diferencia de IP multidifusión, donde la función de multidifusión es realizada por los routers en la capa IP, las redes solapadas llevan a cabo la multidifusión en la capa de aplicación. Por ejemplo, el host de origen puede enviar el flujo a tres servidores solapados; cada uno de los servidores solapados puede reenviar el flujo a los demás servidores y hosts solapados; el proceso continúa, creando un árbol de distribución por encima de la red IP subyacente. Transmitiendo por multidifusión el tráfico en vivo más popular a través de redes solapadas, la carga de tráfico global de Internet puede reducirse respecto al caso de la distribución por unidifusión.

Entre el grupo que defiende las reservas y el que defiende el *laissez-faire* existe un tercer grupo, el que defiende los servicios diferenciados (*Diffserv*). Este grupo desea realizar cambios relativamente pequeños en las capas de red y de transporte e introducir esquemas simples de políticas y de precios en la frontera de la red (es decir, en la interfaz entre el usua-

rio y el ISP del usuario). La idea consiste en introducir un número pequeño de clases de tráfico (posiblemente sólo dos clases), asignar cada datagrama a una de las clases, asignar a cada datagrama un nivel de servicio diferente dependiendo de su clase en las colas de los routers y cobrar a los usuarios de acuerdo con la clase de paquetes que esté enviando a la red. En la Sección 7.5 nos ocuparemos de los servicios diferenciados.

Estos tres enfoques para el tratamiento del tráfico multimedia (mejorar el servicio de mejor esfuerzo, QoS diferencial y QoS garantizado) se resumen en la Tabla 7.1 y se abordan en las Secciones 7.3, 7.5 y 7.6, respectivamente.

7.1.4 Compresión de audio y vídeo

Antes de poder transmitir a una red de computadoras el audio y el vídeo es necesario digitalizarlo y comprimirlo. La necesidad de la digitalización es obvia: las redes de computadoras transmiten bits, de modo que toda la información transmitida tiene que representarse como una secuencia de bits. La compresión es importante porque el audio y el vídeo descomprimidos consumen enormes cantidades de almacenamiento y de ancho de banda: eliminando las redundancias inherentes mediante la compresión de las señales de audio y vídeo digitalizadas puede reducirse la cantidad de datos que habrá que almacenar y transmitir en varios órdenes de magnitud. Por ejemplo, una única imagen de 1024 píxeles, con cada píxel codificado con 24 bits (8 bits por cada uno de los colores rojo, verde y azul) requiere 3 Mbytes de almacenamiento sin compresión. Se tardarían siete minutos en enviar esta imagen a través de un enlace de 64 kbps. Si la imagen se comprime con una relación de comprensión modesta de 10:1, el requisito de almacenamiento se reduce a 300 kbytes y el tiempo de transmisión se reduce también en un factor de 10.

El tema de la compresión del audio y el vídeo es muy extenso, ya que ha constituido un área de investigación durante más de 50 años y actualmente existen, literalmente, cientos de técnicas y estándares populares para la compresión de audio y de vídeo. Muchas universidades ofrecen cursos completos dedicados a la compresión de audio y la compresión de vídeo. Por tanto, aquí sólo vamos a proporcionar una breve introducción de carácter general sobre el tema.

Enfoque	Unidad de asignación	Garantía	Implantación hasta la fecha	Complejidad	Mecanismos
Mejorar el servicio de mejor esfuerzo	ninguna	ninguna o parcial	en cualquier lugar	mínima	soporte de la capa de aplicación, CDN, sobredimensionamiento
QoS diferencial	clases de flujos	ninguna o parcial	alguna	media	vigilancia, planificación
QoS garantizado	flujos individuales	parcial o estricta, una vez que un flujo es admitido	poca	alta	vigilancia, planificación, admisión y señalización de llamadas

Tabla 7.1 • Tres métodos para dar soporte a las aplicaciones multimedia.

Compresión de audio en Internet

Una señal de audio analógica que varía de forma continua (que podría ser voz o música) normalmente se convierte en una señal digital del siguiente modo:

- En primer lugar, la señal de audio analógica se muestrea a una tasa fija, por ejemplo, a 8.000 muestras por segundo. El valor de cada muestra es un número real arbitrario.
- A continuación, cada una de las muestras se redondea a uno de un número finito de valores. Esta operación se conoce con el nombre de **cuantización**. El número de valores finitos (denominados valores de cuantización) normalmente es una potencia de dos, por ejemplo, 256 valores de cuantización.
- Cada uno de los valores de cuantización se representa mediante un número fijo de bits. Por ejemplo, si hay 256 valores de cuantización, entonces cada valor (y por tanto cada muestra) se representa mediante 1 byte. Cada una de las muestras se convierte a su representación de bits. Las representaciones de bits de todas las muestras se concatenan para formar la representación digital de la señal.

Por ejemplo, si una señal de audio analógica se muestrea con una tasa de 8.000 muestras por segundo y cada muestra se cuantiza y representa mediante 8 bits, entonces la señal digital resultante tendrá una tasa de 64.000 bits por segundo. Esta señal digital puede entonces convertirse de nuevo (es decir, decodificarse) en una señal analógica para su reproducción. Sin embargo, normalmente la señal analógica decodificada es diferente de la señal de audio original. Aumentando la tasa de muestreo y el número de valores de cuantización, la señal decodificada puede aproximarse a la señal analógica original. Por tanto, existe una clara relación de compromiso entre la calidad de la señal decodificada y los requisitos de almacenamiento y ancho de banda de la señal digital.

La técnica de codificación básica que acabamos de describir se conoce como **Modulación por código de pulsos (PCM, *Pulse Code Modulation*)**. La codificación de voz normalmente utiliza PCM, con una tasa de muestreo de 8.000 muestras por segundo y 8 bits por muestra, lo que proporciona una tasa de 64 kbps. Los discos de audio compacto (CD) también emplean la modulación PCM, con una tasa de muestreo de 44.100 muestras por segundo y 16 bits por muestra; esto proporciona una tasa de 705,6 kbps para mono y de 1,411 Mbps para estéreo.

Una tasa de bit de 1,411 Mbps para música en estéreo excede la mayoría de las velocidades de acceso e incluso los 64 kbps para voz exceden la velocidad de acceso de un usuario de módem de acceso telefónico. Por estas razones, la voz y la música codificadas mediante PCM rara vez se utilizan en Internet; en su lugar se emplean técnicas de compresión para reducir las tasas de bit de los flujos. Entre las técnicas de compresión de voz populares se incluyen **GSM** (13 kbps), **G.729** (8 kbps), **G.723.3** (6,4 y 5,3 kbps) y una gran cantidad de técnicas propietarias. Una técnica de compresión popular para música estéreo de calidad tipo CD es **MPEG 1 layer 3**, más comúnmente conocida como **MP3**. Los codificadores de MP3 normalmente comprimen a tasas de 96 kbps, 128 kbps y 160 kbps y degradan muy poco el sonido. Cuando un archivo MP3 se divide en fragmentos, cada fragmento sigue siendo reproducible. Este formato de archivo sin cabecera permite transmitir a través de Internet los archivos de música MP3 como si fueran flujos (suponiendo que la velocidad de bit de reproducción y la velocidad de la conexión a Internet sean compatibles). El estándar de compresión MP3 es complejo y utiliza mecanismos de enmascaramiento psicoacústico, reducción de redundancia y buffers de reserva de bits.

Compresión de vídeo en Internet

Un vídeo es una secuencia de imágenes, normalmente reproducidas a una velocidad constante; por ejemplo, a 24 o 30 imágenes por segundo. Una imagen codificada digitalmente y no comprimida consta de una matriz de píxeles, con cada píxel codificado mediante una serie de bits que representan la luminancia y el color. Existen dos tipos de redundancia en el vídeo que pueden ser explotados por los mecanismos de compresión. La redundancia espacial es la redundancia contenida en la propia imagen. Por ejemplo, una imagen que tiene gran cantidad de espacio en blanco puede ser comprimida de forma eficiente. La redundancia temporal refleja la repetición de una imagen a la siguiente. Por ejemplo, si una imagen y la siguiente son exactamente iguales, no existe ninguna razón para codificar también la segunda de ellas; es más eficiente indicar simplemente durante el proceso de codificación que la segunda imagen es exactamente igual que la primera.

Los estándares de compresión MPEG se encuentran entre las técnicas de compresión más populares. Entre estos se incluyen **MPEG 1** para vídeo de calidad CD-ROM (1,5 Mbps), **MPEG 2** para vídeo de alta calidad **DVD** (3-6 Mbps) y **MPEG 4** para compresión de vídeo orientada a objeto. Los estándares MPEG están basados fundamentalmente en el estándar JPEG para la compresión de imágenes y explotan la redundancia temporal de las imágenes, además de la redundancia espacial explotada por JPEG. Los estándares de compresión de vídeo **H.261** también son muy populares en Internet. Además, existen numerosos esquemas propietarios, entre los que se incluyen QuickTime de Apple y los codificadores Real Networks.

Los lectores interesados en aprender más acerca de la codificación de audio y vídeo pueden consultar [Rao 1996] y [Solari 1997]. Un buen libro sobre las redes multimedia en general es [Crowcroft 1999].

7.2 Flujos de audio y de vídeo almacenado

En los últimos años, la transmisión de flujos de audio y de vídeo se ha convertido en una aplicación popular y en un consumidor importante de ancho de banda de red. En estas aplicaciones, el cliente solicita archivos comprimidos de audio/vídeo que residen en servidores. Como pronto veremos, estos servidores pueden ser servidores web normales o servidores especiales para flujos adaptados para aplicaciones de flujos de audio/vídeo. Una vez recibida la solicitud del cliente, el servidor envía un archivo de audio/vídeo al cliente pasando el archivo a un socket. Aunque se puede utilizar tanto TCP como UDP, hoy día la mayor parte del tráfico de flujos de audio/vídeo es transportado por TCP. (Los cortafuegos a menudo se configuran para bloquear el tráfico de UDP. Además, utilizando TCP, con su servicio de entrega fiable, el archivo completo consigue transferirse al cliente sin pérdida de paquetes, permitiendo que el archivo pueda ser reproducido en el futuro desde una caché local.) [Sripanidkulchai 2004]. Una vez que el archivo de audio/vídeo solicitado comienza a llegar, el cliente inicia su reproducción pasados unos pocos segundos. Algunos sistemas también proporcionan interactividad con el usuario; por ejemplo, el usuario puede pausar/reanudar el archivo de audio/vídeo o saltar de un punto a otro dentro del mismo. El **protocolo de transmisión de flujos en tiempo real (RTSP, *Real-Time Streaming Protocol*)**, que veremos al final de esta sección, es un protocolo de dominio público que proporciona interactividad con el usuario.



HISTORIA

FLUJOS DE AUDIO Y DE VÍDEO ALMACENADO: DESDE REALNETWORKS A YOUTUBE

RealNetworks, una empresa pionera en la transmisión de flujos de audio y de vídeo, fue la primera compañía que ofreció audio a través de Internet al público en general. Su producto inicial (el sistema RealAudio distribuido en 1995) incluía un codificador de audio, un servidor de audio y un reproductor de audio. Permitía a los usuarios navegar, seleccionar y reproducir contenido de audio desde Internet bajo demanda, por lo que rápidamente se convirtió en un sistema de distribución popular para los proveedores de contenido de entretenimiento, educativo y de noticias.

Actualmente, los flujos de audio y vídeo se encuentran entre los servicios más populares de Internet. No sólo existe una gran cantidad de empresas que ofrecen contenido de este tipo, sino que también se están empleando una multitud de distintos servidores, reproductores y tecnologías de protocolo. Algunos ejemplos interesantes (a fecha de 2009) son:

- **Rhapsody de RealNetworks:** proporciona a los usuarios servicios de suscripción para descarga y transmisión de flujos. Rhapsody utiliza su propio cliente propietario, que recupera canciones de su servidor propietario utilizando HTTP. Cuando una canción llega a través de HTTP se reproduce en su cliente Rhapsody. El acceso al contenido descargado está restringido mediante un sistema de Gestión de derechos digital (DRM, *Digital Rights Management*).
- **MSN Video:** los usuarios se descargan una amplia variedad de contenidos, incluyendo noticias internacionales y clips de vídeos musicales. El vídeo se reproduce mediante el popular reproductor Windows Media Player (WMP), que está disponible en casi todos los hosts Windows. La comunicación entre WMP y el servidor de Microsoft se hace mediante el protocolo propietario MMS (Microsoft Media Server), que normalmente intenta transmitir los flujos sobre RTSP/RTP; si esta solución falla a causa de los cortafuegos, prueba a recuperar el contenido a través de HTTP.
- **Muze:** proporciona un servicio de muestras de audio para minoristas, como BestBuy y Yahoo. Las muestras de música seleccionadas en los sitios de estos minoristas realmente proceden de Muze, y se transmiten a través de WMP. Muze, Rhapsody, YouTube y muchos otros proveedores de flujos multimedia utilizan redes de distribución de contenido (CDN) para distribuir sus contenidos, como se explica en la Sección 7.3.
- **YouTube:** el tremendamente popular servicio de compartición de vídeos utiliza un cliente basado en Flash (integrado en la página web). La comunicación entre el cliente y los servidores de YouTube se hace a través de HTTP.

¿Qué es lo que nos deparará el futuro? Actualmente, la mayor parte de los flujos de vídeo son de baja calidad y se codifican a tasas de 500 kbps o menores. La calidad del vídeo realmente mejorará cuando el acceso a Internet de banda ancha y de fibra en los domicilios adquiera una mayor predominancia. Y muy posiblemente nuestros reproductores portátiles de música ya no almacenarán la música, sino que la obtendrán directamente, bajo demanda, a través de canales inalámbricos.

Los usuarios a menudo solicitan la transmisión de flujos de audio/vídeo a través de un cliente web (es decir, un navegador), pero luego reproducen y controlan la reproducción de éstos mediante un **reproductor multimedia**, como por ejemplo Windows Media Player o Flash. El reproductor multimedia realiza varias funciones, entre las que se incluyen las siguientes:

- *Descompresión.* El audio/vídeo casi siempre se comprime con el fin de ahorrar espacio de almacenamiento en disco y ancho de banda de red. Un reproductor multimedia tiene que descomprimir el audio/vídeo sobre la marcha durante la reproducción.
- *Eliminación de las fluctuaciones.* La fluctuación de los paquetes es la variabilidad de los retardos del origen al destino de los paquetes contenidos en un mismo flujo de paquetes. Puesto que el audio y el vídeo tienen que reproducirse con la misma temporización que fueron grabados, un receptor almacenará en buffer los paquetes recibidos durante un corto periodo de tiempo para eliminar esa fluctuación. Examinaremos este tema en detalle en la Sección 7.3.

7.2.1 Acceso al audio y al vídeo a través de un servidor web

El audio/vídeo almacenado puede residir bien en un servidor web que entrega los archivos de audio/vídeo al cliente a través de HTTP, o bien en un servidor dedicado de flujos de audio/vídeo utilizando HTTP o algún otro protocolo. En esta subsección, vamos examinar la entrega de flujos de audio/vídeo desde un servidor web; en la siguiente subsección nos centraremos en la entrega desde un servidor de flujos. El suministro de flujos multimedia vía HTTP se ha hecho muy popular gracias a los cortafuegos (véase el Capítulo 8), que suelen permitir que el tráfico HTTP los atraviese, mientras que los protocolos propietarios son bloqueados.

Consideremos en primer lugar el caso de los flujos de audio. Cuando un archivo de audio reside en un servidor web, el archivo es un objeto ordinario dentro del sistema de archivos del servidor, como lo son los archivos HTML y JPEG. Cuando un usuario desea escuchar el archivo de audio, el host del usuario establece una conexión TCP con el servidor web y envía una solicitud HTTP para el objeto. Después de recibir una solicitud, el servidor web encapsula el archivo de audio en un mensaje de respuesta HTTP y lo envía de vuelta a través de la conexión TCP. El caso de un archivo de vídeo puede ser un poco más complicado si las partes de audio y vídeo están almacenadas en dos archivos. También es posible que el audio y el vídeo vayan intercalados en el mismo archivo, en cuyo caso sólo habrá que enviar un objeto al cliente. Con el fin de simplificar nuestra exposición, vamos a suponer que en el caso del vídeo, ambas partes, audio y vídeo, están contenidas en un único archivo.

En muchas implementaciones de transmisión de flujos de audio/vídeo sobre HTTP, la funcionalidad del lado del cliente se divide en dos partes. La tarea del navegador es solicitar un **metarchivo** que proporciona información (por ejemplo, un URL y un tipo de codificación, de modo que pueda ser identificado el reproductor multimedia apropiado) acerca del archivo multimedia que va a ser transmitido mediante HTTP. Este metarchivo se pasa entonces desde el navegador al reproductor multimedia, cuyo trabajo consiste en contactar al servidor HTTP, el cual a continuación envía el archivo multimedia al reproductor vía HTTP. Estos pasos se ilustran en la Figura 7.1:

1. El usuario hace clic en el hipervínculo correspondiente a un archivo de audio/vídeo. El hipervínculo no apunta directamente a dicho archivo, sino a un metarchivo. El metar-

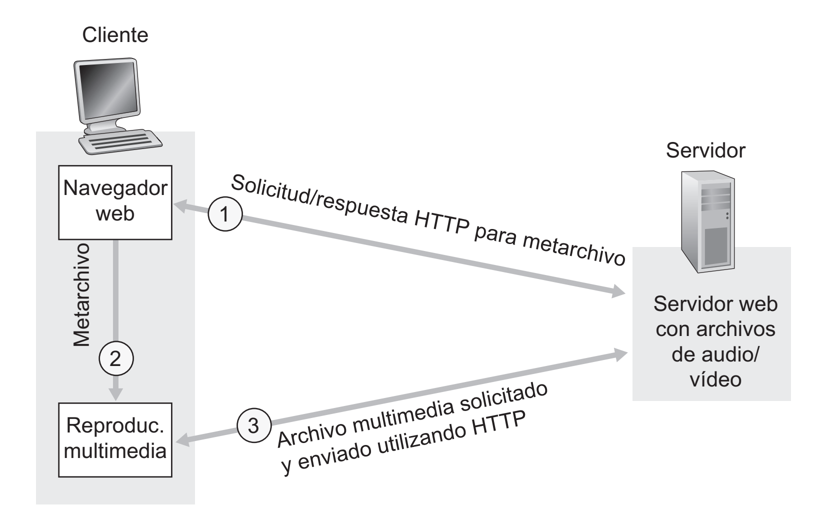


Figura 7.1 • El servidor web envía directamente el flujo de audio/vídeo al reproductor multimedia.

chivo contiene el URL al archivo de audio/vídeo real. El mensaje de respuesta HTTP que encapsula el metarchivo incluye una línea de cabecera con el tipo de contenido que indica una aplicación de audio/vídeo específica.

2. El navegador del cliente examina la línea de cabecera correspondiente al tipo de contenido del mensaje de respuesta, ejecuta el reproductor multimedia asociado y pasa el cuerpo del mensaje de respuesta (es decir, el metarchivo) al reproductor.
3. El reproductor multimedia establece una conexión TCP directamente con el servidor HTTP. El reproductor multimedia envía un mensaje de respuesta HTTP para el archivo de audio/vídeo a través de la conexión TCP. El archivo de audio/vídeo se envía dentro de un mensaje de respuesta HTTP hacia el reproductor, el cual reproduce el archivo.

La importancia del paso intermedio de adquisición del metarchivo es evidente. Cuando el navegador ve el tipo de contenido del archivo puede ejecutar el reproductor apropiado y, de este modo, hacer que el reproductor contacte directamente con el servidor.

Acabamos de ver cómo un metarchivo puede hacer que un reproductor se comunique directamente con un servidor web que almacena un archivo de audio/vídeo. Existen todavía muchas empresas que venden productos para la transmisión de flujos de audio/vídeo que no recomiendan la arquitectura que acabamos de describir. En su lugar, recomiendan transmitir los flujos de audio/vídeo almacenado desde servidores de flujos dedicados, los cuales han sido optimizados para dicha tarea.

7.2.2 Envío de información multimedia desde un servidor de flujos a una aplicación de ayuda

Un servidor de flujos puede ser un servidor de flujos propietario, como los comercializados por RealNetworks y Microsoft, o puede ser un servidor de flujos de dominio público. Con un servidor de flujos, el audio/vídeo puede enviarse a través de HTTP/TCP; o mediante

UDP utilizando protocolos de la capa de aplicación que pueden adaptarse mejor que HTTP a la transmisión de flujos de audio/vídeo.

Esta arquitectura requiere dos servidores, como se muestra en la Figura 7.2. Un servidor, el servidor web, sirve las páginas web (incluyendo metarchivos). El segundo servidor, el **servidor de flujos**, sirve los archivos de audio/vídeo. Los dos servidores pueden ejecutarse en el mismo sistema terminal o en dos sistemas terminales distintos. Los pasos que se siguen en esta arquitectura son similares a los descritos en la anterior subsección. Sin embargo, ahora el reproductor multimedia solicita el archivo a un servidor de flujos en lugar de a un servidor web, y ahora el reproductor y el servidor de flujos pueden interactuar utilizando sus propios protocolos. Estos protocolos permiten una rica interacción del usuario con el flujo de audio/vídeo.

En la arquitectura de la Figura 7.2 existen muchas opciones para la entrega del audio/vídeo desde el servidor de flujos al reproductor multimedia. A continuación proporcionamos una lista parcial de las opciones.

1. El archivo de audio/vídeo se envía sobre UDP a una velocidad constante igual a la velocidad de vaciado del receptor (que es la tasa de codificación del audio/vídeo). Por ejemplo, si el audio se comprime utilizando GSM a una tasa de 13 kbps, entonces el servidor temporiza la transmisión del archivo de audio comprimido a 13 kbps. En cuanto el cliente recibe el audio/vídeo comprimido de la red, lo descomprime y reproduce.
2. Esto es lo mismo que en la primera opción, pero en este caso el reproductor multimedia retarda la reproducción entre dos y cinco segundos con el fin de eliminar la fluctuación inducida por la red. El cliente lleva a cabo esta tarea insertando el archivo comprimido que recibe de la red en un **buffer cliente**, como se muestra en la Figura

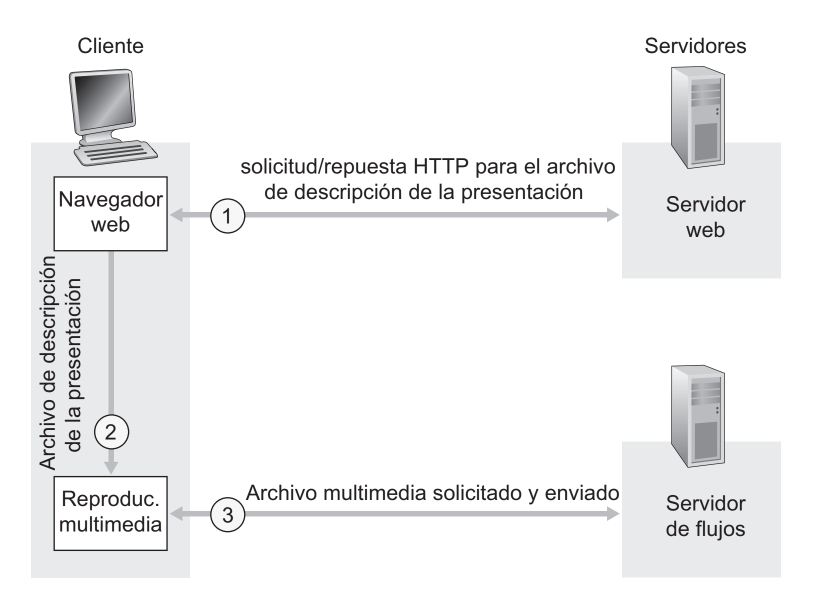


Figura 7.2 • Transmisión desde un servidor de flujos a un reproductor multimedia.

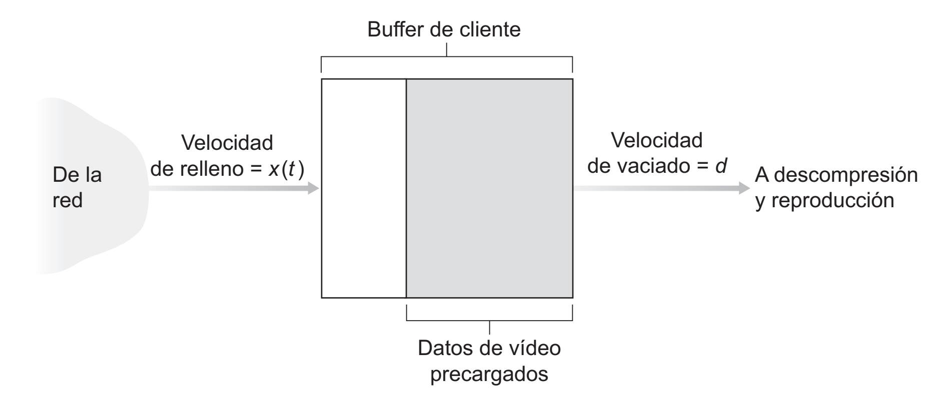


Figura 7.3 • El buffer cliente se llena a una velocidad $x(t)$ y se vacía a una velocidad d.

7.3. Una vez que el cliente ha precargado unos pocos segundos del archivo comienza a vaciar el buffer. En este caso, y también en la opción anterior, la velocidad de relleno $x(t)$ es igual a la velocidad de vaciado d, excepto cuando se produce una pérdida de paquetes, en cuyo caso $x(t)$ será momentáneamente menor que d.

3. Los archivos multimedia se envían sobre TCP. El servidor inserta el archivo multimedia en el socket TCP tan rápidamente como puede; el cliente (es decir, el reproductor multimedia) lee del socket TCP tan rápido como puede y coloca el vídeo comprimido en el buffer del reproductor. Después de un retardo inicial de dos a cinco segundos, el reproductor lee de su buffer a una velocidad d y reenvía el archivo comprimido a descompresión y reproducción. Dado que TCP retransmite los paquetes perdidos, tiene el potencial de proporcionar una mejor calidad de sonido que UDP. Por otro lado, la velocidad de relleno $x(t)$ ahora fluctúa con el tiempo a causa del control de congestión y del control de flujo de ventana de TCP. En realidad, después de producirse una pérdida de paquetes, el control de congestión de TCP puede reducir la velocidad instantánea a un valor menor que el de d durante largos periodos de tiempo. De este modo, el buffer cliente puede vaciarse (un proceso conocido como **inanición**) e introducir pausas indeseables en la salida del flujo de audio/vídeo en el cliente. [Wang 2004] demuestra que cuando la tasa media de transferencia de TCP es aproximadamente igual a dos veces la velocidad media de bit, los flujos TCP permiten minimizar la inanición y obtener retardos de arranque pequeños.

Para la tercera opción, el comportamiento de $x(t)$ dependerá en gran medida del tamaño del buffer cliente (el cual no debe confundirse con el buffer del receptor TCP). Si este buffer es lo suficientemente grande como para almacenar el archivo multimedia completo (posiblemente en disco), entonces TCP hará uso de todo el ancho de banda instantáneo disponible para la conexión, de modo que $x(t)$ pueda hacerse mucho mayor que d. Si $x(t)$ es mucho mayor que d durante largos periodos de tiempo, entonces gran parte del archivo multimedia se precarga en el cliente y la subsiguiente inanición del cliente es muy improbable. Si, por el contrario, el buffer del cliente es pequeño, entonces $x(t)$ fluctuará alrededor del valor de la velocidad de vaciado d, siendo entonces el riesgo de inanición del cliente mucho mayor.

7.2.3 Protocolo de transmisión de flujos en tiempo real (RTSP)

Muchos usuarios multimedia de Internet (especialmente aquellos que han crecido con el mando de la TV en la mano) desearán controlar la reproducción de los medios continuos pausando la reproducción, saltando a un punto anterior o posterior de la misma, haciendo un avance rápido o un rebobinado visual de la reproducción, etc. Esta funcionalidad es similar a la de la que dispone un usuario que tiene un reproductor de DVD para la visualización de vídeos en DVD o un reproductor de discos CD para escuchar música. Para que un usuario pueda controlar la reproducción, el reproductor multimedia y el servidor necesitan un protocolo que les permita intercambiar la información de control de la reproducción. El Protocolo de flujos en tiempo real (RTSP, *Real-Time Streaming Protocol*), definido en RFC 2326, es dicho protocolo. Antes de abordar los detalles de RTSP, veamos primero qué no hace RTSP.

- RTSP no define esquemas de compresión para audio y vídeo.
- RTSP no define cómo se encapsulan el audio y el vídeo en paquetes para su transmisión a través de una red; RTP o cualquier otro protocolo propietario puede proporcionar el mecanismo de encapsulación para los flujos multimedia (RTP se estudia en la Sección 7.4). Por ejemplo, los servidores y reproductores de audio/vídeo de RealNetworks utilizan RTSP para intercambiar información de control, pero el propio flujo multimedia puede ser encapsulado en paquetes RTP o en algún otro formato de datos propietario.
- RTSP no restringe cómo se transporta el flujo multimedia; puede ser transportado sobre UDP o TCP.
- RTSP no restringe cómo el reproductor multimedia almacena en buffer el audio/vídeo. El flujo de audio/vídeo puede reproducirse tan pronto como empieza a llegar al cliente, puede reproducirse después de un retardo de unos pocos segundos o puede descargarse completo antes de iniciar la reproducción.

Por tanto, si RTSP no hace nada de lo anterior, ¿qué hace? RTSP permite que un reproductor multimedia controle la transmisión de un flujo multimedia. Como ya hemos mencionado, las acciones de control son: pausar/reanudar, reposicionar la reproducción, avance rápido y rebobinado. RTSP es un **protocolo fuera de banda**. En particular, los mensajes RTSP se envían fuera de banda, mientras que los flujos multimedia, cuya estructura de paquete no está definida por RTSP, se consideran “en banda”. Los mensajes RTSP utilizan un número de puerto diferente, el 544, del empleado por los flujos multimedia. La especificación RTSP [RFC 2326] permite que los mensajes RTSP sean enviados sobre TCP o sobre UDP.

Recuerde de la Sección 2.3 que el Protocolo de transferencia de archivos (FTP) también utiliza el concepto de fuera de banda. En particular, FTP emplea dos parejas de sockets cliente/ servidor, teniendo cada pareja su propio número de puerto: una pareja de sockets cliente/servidor soporta una conexión TCP que transporta información de control y la otra pareja de sockets cliente/servidor soporta una conexión TCP que realmente transporta el archivo. El canal de RTSP es similar en muchos puntos al canal de control de FTP.

Veamos ahora un ejemplo simple de RTSP, el cual se ilustra en la Figura 7.4. En primer lugar, el navegador web solicita de un servidor web un archivo de descripción de la presentación. Este archivo de descripción puede contener referencias a varios archivos multimedia continuos, así como directivas para la sincronización de los mismos. Cada referencia a un archivo multimedia continuo comienza con el método URL, `rtsp://`. A continuación proporcionamos un archivo de presentación que ha sido adaptado de [Schulzrinne 1997]. En

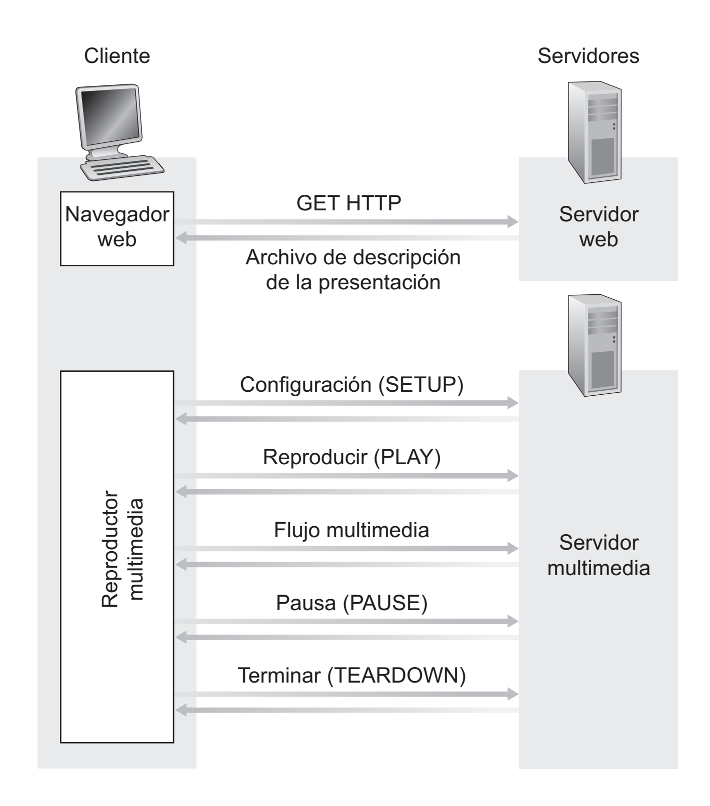


Figura 7.4 • Interacción entre un cliente y un servidor utilizando RTSP.

esta presentación se reproducen sendos flujos de audio y vídeo en paralelo y con sincronización del movimiento de los labios (como parte del mismo grupo). Para el flujo de audio, el reproductor multimedia puede elegir (conmutar) entre dos grabaciones de audio, una de baja fidelidad y otra de alta fidelidad. (El formato del archivo es similar a SMIL [SMIL 2009], formato que emplean muchos productos de transmisión de flujos multimedia para definir presentaciones multimedia sincronizadas.)

El servidor web encapsula el archivo de descripción de la presentación en un mensaje de respuesta HTTP y lo envía al navegador. Cuando el navegador recibe dicha respuesta HTTP, invoca a un reproductor multimedia (es decir, a la aplicación de ayuda) basándose en el campo que define el tipo de contenido del mensaje. El archivo de descripción de la presentación hace referencia a los flujos multimedia utilizando el método URL `rtsp://`, al igual que en el ejemplo anterior. Como se muestra en la Figura 7.4, el reproductor y el servidor intercambian una serie de mensajes RTSP. El reproductor envía una solicitud RTSP SETUP y el servidor responde con una mensaje RTSP OK. El reproductor envía una solicitud RTSP de reproducción PLA de, digamos, audio de baja fidelidad y el servidor le responde con un mensaje RTSP OK. En este punto, el servidor de flujos coloca el audio de baja fidelidad en su propio canal en banda. A continuación, el reproductor multimedia envía una solicitud RTSP de pausa con el mensaje PAUSE y el servidor responde de nuevo con un mensaje RTSP OK. Cuando el usuario ha terminado, el reproductor multimedia envía una solicitud RTSP TEARDOWN para terminar la sesión y el servidor confirma con un mensaje de respuesta RTSP OK.

```
<title>Twister</title>
<session>
      <group language=en lipsync>
         <switch>
            <track type=audio
                  e="PCMU/8000/1"
                  src="rtsp://audio.example.com/twister/audio.en/lofi">
            <track type=audio
                  e="DVI4/16000/2" pt="90 DVI4/8000/1"
                  src="rtsp://audio.example.com/twister/audio.en/hifi">
               </switch>
            <track type="video/jpeg"
                  src="rtsp://video.example.com/twister/video">
      </group>
</session>
```

Veamos ahora los mensajes RTSP reales. El siguiente listado es un ejemplo simplificado de una sesión RTSP entre un cliente (C:) y un servidor (S:).

```
C:    SETUP rtsp://audio.example.com/twister/audio RTSP/1.0
      Cseq: 1
      Transport: rtp/udp; compression; port=3056; mode=PLAY
S:    RTSP/1.0 200 OK
      Cseq: 1
      Session: 4231
C:    PLAY rtsp://audio.example.com/twister/audio.en/lofi RTSP/1.0
      Range: npt=0-
      Cseq: 2
      Session: 4231
S:    RTSP/1.0 200 OK
      Cseq: 2
      Session: 4231
C:    PAUSE rtsp://audio.example.com/twister/audio.en/lofi RTSP/1.0
      Range: npt=37
      Cseq: 3
      Session: 4231
S:    RTSP/1.0 200 OK
      Cseq: 3
      Session: 4231
C:    TEARDOWN rtsp://audio.example.com/twister/audio.en/lofi RTSP/1.0
      Cseq: 4
      Session: 4231
S:    RTSP/1.0 200 OK
      Cseq: 4
      Session: 4231
```

Es interesante destacar las similitudes entre HTTP y RTSP. Todos los mensajes de solicitud y de respuesta están en texto ASCII, el cliente emplea métodos estandarizados

(SETUP, PLAY, PAUSE, etc.) y el servidor responde con códigos de respuesta estandarizados. No obstante, una diferencia importante es que el servidor RTSP controla el estado del cliente para cada sesión RTSP activa. Por ejemplo, el servidor controla si el cliente se encuentra en el estado de inicialización, de reproducción o en pausa (véase la tarea de programación al final del capítulo). La sesión y los números de secuencia, que forman parte de cada mensaje de solicitud y de respuesta RTSP, ayudan al servidor a controlar el estado de la sesión. El número de sesión se fija para la sesión completa; el cliente incrementa el número de secuencia cada vez que envía un nuevo mensaje; el servidor se hace eco del número de sesión y del número de secuencia actual.

Como se muestra en el ejemplo, el cliente inicia la sesión con la solicitud SETUP, proporcionando el URL del archivo que va a ser transmitido como flujo y la versión de RTSP. El mensaje de configuración (SETUP) incluye el número de puerto del cliente al que debe enviarse el archivo multimedia. Este mensaje de configuración también indica que el archivo deberá ser enviado sobre UDP utilizando el protocolo de empaquetamiento RTP (que veremos en la Sección 7.4). Observe que, en este ejemplo, el reproductor elige no reproducir la presentación completa, sino sólo la parte de baja fidelidad de la misma.

RTSP realmente es capaz de hacer mucho más que lo que hemos explicado en esta breve introducción. En particular, RTSP dispone de facilidades que permiten a los clientes dirigir flujos hacia el servidor (por ejemplo, para grabar). RTSP ha sido adoptado por RealNetworks, uno de los líderes de la industria en flujos de audio/vídeo. Henning Schulzrinne proporciona una página web sobre RTSP [Schulzrinne-RTSP 2009].

Al final del capítulo puede encontrar una tarea de programación para crear un sistema de flujos de vídeo (tanto de servidor como de cliente) que se aprovecha de RTSP. Esta tarea precisa la escritura de código que construya y envíe realmente mensajes RTSP en el cliente. La tarea proporciona el código de servidor RTSP, el cual analiza los mensajes RTSP y construye las respuestas apropiadas. A los lectores interesados en obtener unos conocimientos más profundos sobre RTSP les animamos a que lleven a cabo esta interesante tarea.

7.3 Utilización óptima del servicio de entrega de mejor esfuerzo

El protocolo Internet de la capa de red, IP, proporciona un servicio de entrega de mejor esfuerzo. Esto quiere decir que el servicio trata de hacer lo que puede para transferir cada datagrama desde el origen hasta el destino de la forma más rápida posible. Sin embargo, no realiza ningún tipo de promesa en lo que se refiere a la duración del retardo terminal a terminal para un paquete individual, ni de la intensidad de las fluctuaciones temporales de los paquetes, ni tampoco acerca de las pérdidas de paquetes dentro del flujo de datos. La falta de garantías acerca del retardo y de la fluctuación de los paquetes impone una serie de desafíos significativos al diseño de aplicaciones multimedia en tiempo real, como la telefonía por Internet y las videoconferencias en tiempo real que son enormemente sensibles a los retardos, a las fluctuaciones y a la pérdida de paquetes.

En esta sección vamos a ver diversas formas en las que pueden mejorarse las prestaciones de las aplicaciones multimedia sobre una red basada en un servicio de entrega de mejor esfuerzo. Nos vamos centrar en las técnicas de la capa de aplicación; es decir, en técnicas que no requieren efectuar ninguna modificación en el núcleo de la red, ni tampoco en la capa

de transporte de los hosts terminales. En primer lugar describiremos los efectos de la pérdida de paquetes, de los retardos y de las fluctuaciones de esos retardos sobre las aplicaciones multimedia. A continuación pasaremos a analizar las técnicas utilizadas para sobreponerse a esas deficiencias. Después describiremos cómo pueden emplearse las redes de distribución de contenido y las técnicas de sobredimensionamiento de recursos para evitar la aparición de esas deficiencias.

7.3.1 Limitaciones de un servicio de entrega de mejor esfuerzo

Hemos mencionado que el servicio de entrega de mejor esfuerzo puede conducir a la pérdida de paquetes, a que se experimente un retardo excesivo terminal a terminal y a que se manifieste una alta fluctuación temporal de los paquetes. Vamos a examinar estas cuestiones más en detalle. Para que nuestro análisis sea lo más concreto posible, vamos a hablar de estos mecanismos en el contexto de una **aplicación de telefonía Internet**, descrita más abajo. La situación es similar para las aplicaciones de videoconferencia en tiempo real [Bolot 1994].

El hablante en nuestro ejemplo de telefonía por Internet genera una señal de audio que está compuesta de una serie alternativa de periodos de conversación y periodos de silencio. Para poder ahorrar ancho de banda, nuestra aplicación de telefonía por Internet sólo genera paquetes durante los periodos de conversación. En cada uno de esos periodos de conversación, el emisor genera bytes a una velocidad de 8.000 bytes por segundo y cada 20 milisegundos agrupa esos bytes en una serie de fragmentos. Por tanto, el número de bytes en un fragmento será igual a (20 milisegundos) · (8.000 bytes/segundo) = 160 bytes. A cada fragmento se le asocia una cabecera especial cuyo contenido analizaremos más adelante. El fragmento y su cabecera se encapsulan en un segmento UDP mediante una llamada a la interfaz de socket. De este modo, durante un periodo de conversación se envía un segmento UDP cada 20 milisegundos.

Si cada paquete consigue llegar al receptor y tiene un retardo terminal a terminal de duración constante y pequeña, entonces los paquetes llegarán al receptor periódicamente cada 20 milisegundos durante los periodos de conversación. En estas condiciones ideales, el receptor puede simplemente reproducir cada fragmento en cuanto lo recibe. Pero lamentablemente, algunos paquetes pueden perderse y la mayoría de los paquetes no tendrá el mismo retardo terminal a terminal, incluso en una Internet que esté congestionada sólo ligeramente. Por esta razón, el receptor debe tener algo más de cuidado a la hora de determinar (1) cuándo hay que reproducir un fragmento y (2) qué hay que hacer cuando falta un fragmento.

Pérdida de paquetes

Considere uno de los segmentos UDP generados por nuestra aplicación de telefonía Internet. El segmento UDP se encapsula dentro de un datagrama IP. A medida que el datagrama viaja por la red pasa por una serie de buffers (es decir, colas) en los routers para poder acceder a los enlaces de salida de éstos. Es posible que uno o más de los buffers en la ruta existente entre el emisor y el receptor esté lleno y no pueda admitir el datagrama IP. En este caso el datagrama IP será descartado, por lo que nunca llegará a la aplicación receptora.

Las pérdidas podrían eliminarse enviando los paquetes sobre TCP en lugar de sobre UDP. Recuerde que TCP retransmite los paquetes que no llegan a su destino. Sin embargo, los mecanismos de retransmisión se suelen considerar inaceptables para las aplicaciones de

audio interactivas en tiempo real, como la de telefonía por Internet, porque incrementan el retardo terminal a terminal [Bolot 1996]. Además, debido al control de congestión de TCP, la velocidad de transmisión en el emisor puede verse reducida después de una pérdida de paquetes, fijándose una velocidad inferior a la velocidad de consumo de paquetes en el receptor. Esto puede tener un impacto grave sobre la inteligibilidad de la voz en el receptor. Por estas razones, la mayoría de las aplicaciones de telefonía por Internet existentes se ejecutan sobre UDP y no se preocupan de retransmitir los paquetes perdidos. [Baset 2006] informa de que Skype utiliza UDP a menos que un usuario se encuentre detrás de un traductor NAT o de un cortafuegos que bloquee los segmentos UDP (en cuyo caso se emplea TCP).

Pero las pérdidas de paquetes no son necesariamente tan desastrosas como podría parecer. De hecho, se pueden tolerar perfectamente tasas de pérdida de paquetes de entre el 1 y el 20 por ciento, dependiendo de cómo se codifique y transmita la voz y de cómo se oculten esas pérdidas en el receptor. Por ejemplo, los mecanismos de corrección de errores hacia adelante (FEC, *Forward Error Correction*) pueden ayudar a ocultar las pérdidas de paquetes. Veremos más adelante que con las técnicas FEC se transmite información redundante junto con la información original, de modo que una parte de los datos originales perdidos puede recuperarse a partir de esa información redundante. Sin embargo, si uno o más de los enlaces existentes entre el emisor y el receptor está severamente congestionado y la tasa de pérdida de paquetes excede del 10 o 20 por ciento (aunque estas tasas tan altas raramente se observan en las redes bien diseñadas), entonces no hay nada que podamos hacer para conseguir una calidad de sonido aceptable. Claramente, el servicio de entrega de mejor esfuerzo tiene sus limitaciones.

Retardo terminal a terminal

El **retardo terminal a terminal** es la suma de los retardos de transmisión, de procesamiento y de puesta en cola de los routers; los retardos de propagación de los enlaces y los retardos de procesamiento en los sistemas terminales. Para las aplicaciones de audio altamente interactivas, como la telefonía por Internet, los retardos terminal a terminal inferiores a 150 milisegundos no son perceptibles por los oyentes humanos; los retardos entre 150 y 400 milisegundos son aceptables, aunque distan mucho de ser ideales, y los retardos mayores que 400 milisegundos pueden afectar seriamente a la interactividad en las conversaciones de voz. El lado receptor de una aplicación de telefonía por Internet descartará normalmente todos los paquetes que estén retardados más de un cierto umbral, como por ejemplo más de 400 milisegundos. Por tanto, los paquetes cuyo retardo sea superior al umbral prefijado se perderán.

Fluctuación de los paquetes

Un componente crucial del retardo terminal a terminal son los retardos aleatorios de puesta en cola dentro de los routers. A causa de estos retardos variables dentro de la red, el tiempo que transcurre desde el momento en que se genera un paquete en el origen hasta que se recibe en el destino puede fluctuar de un paquete a otro. Este fenómeno se conoce como **fluctuación** o ***jitter***.

Por ejemplo, considere dos paquetes consecutivos dentro de un periodo de conversación de nuestra aplicación de telefonía por Internet. El emisor envía el segundo paquete 20 milisegundos después de enviar el primero. Pero en el receptor, el espaciado entre estos paquetes puede ser mayor de 20 milisegundos. Para ver por qué, suponga que el primer paquete llega a

una cola prácticamente vacía dentro de un router y que, justo antes de que el segundo paquete llegue a esa misma cola, entran en la cola un gran número de paquetes procedentes de otros orígenes. Puesto que el primer paquete sufre un pequeño retardo de puesta en cola y el segundo paquete sufre un retardo de puesta en cola mucho mayor dentro de este router, el espaciado entre el primer y el segundo paquete será superior a 20 milisegundos. De la misma manera, el espaciado entre paquetes consecutivos también podría ser inferior a 20 milisegundos. Para ver por qué, considere de nuevo dos paquetes consecutivos dentro de un periodo de conversación. Suponga que el primer paquete se coloca al final de una cola que contiene un gran número de paquetes y que el segundo paquete llega a esa cola antes de que otros paquetes de otros orígenes entren en la cola. En este caso, nuestros dos paquetes estarán uno detrás del otro dentro de la cola. Si el tiempo que se tarda en transmitir un paquete a través del enlace de salida del router es inferior a 20 milisegundos, entonces el espaciado entre el primer y el segundo paquete será también inferior a 20 milisegundos.

La situación es análoga a la que se produce cuando circulan vehículos por una carretera. Suponga que usted y un amigo viajan cada uno en su automóvil desde San Diego a Fénix. Suponga también que usted y su amigo tienen formas similares de conducir y que ambos viajan a 100 km/hora cuando el tráfico lo permite. Finalmente, suponga que su amigo comienza el viaje una hora antes que usted. Entonces, dependiendo del tráfico con el que él o usted se encuentren, puede que llegue a Fénix menos de una hora o más de una hora después que su amigo.

Si el receptor ignora la presencia de las fluctuaciones y reproduce los segmentos en cuanto llegan, entonces la calidad del audio resultante puede llegar a ser ininteligible en el receptor. Afortunadamente, las fluctuaciones pueden eliminarse a menudo utilizando **números de secuencia**, **marcas de tiempo** y un **retardo de reproducción**, como se explica a continuación.

7.3.2 Eliminación de las fluctuaciones al reproducir el audio en el receptor

Para una aplicación de voz como la telefonía por Internet o la música bajo demanda, el receptor debe tratar de reproducir síncronamente los fragmentos de voz en presencia de unas fluctuaciones aleatorias en la red; las aplicaciones de vídeo tienen a menudo requisitos similares. Esto normalmente se hace combinando los siguientes tres mecanismos:

- *Precediendo cada fragmento con **un número de secuencia**.* El emisor incrementa el número de secuencia en una unidad por cada uno de los paquetes que genera.
- *Precediendo cada fragmento con **una marca de tiempo**.* El emisor marca cada fragmento con la hora a la que el fragmento fue generado.
- ***Retardando la reproducción** de los fragmentos en el receptor.* El retardo de reproducción de los fragmentos de audio recibidos debe ser lo suficientemente grande como para que la mayor parte de los paquetes se reciban antes de su instante de reproducción planificado. Este retardo de reproducción puede ser fijo a lo largo de toda la sesión de audio, o variar adaptativamente a lo largo de dicha sesión. Los paquetes que no lleguen antes de su tiempo de reproducción planificado se consideran paquetes perdidos y se ignoran; como hemos indicado anteriormente, el receptor puede utilizar algún tipo de interpolación de voz para tratar de ocultar las pérdidas.

Veamos ahora cómo pueden aliviar o incluso eliminar los efectos de las fluctuaciones estos tres mecanismos cuando se les combina apropiadamente. Vamos a examinar dos estrategias de reproducción: el retardo de reproducción fijo y el retardo de reproducción adaptativo.

Retardo de reproducción fijo

Con la estrategia basada en un retardo fijo, el receptor intenta reproducir cada fragmento exactamente q milisegundos después de que ese fragmento haya sido generado. Por tanto, si un fragmento tiene una marca de tiempo que indica que fue generado en el instante t, el receptor reproduce dicho fragmento en el instante $t + q$, suponiendo que el fragmento haya llegado antes de ese momento. Los paquetes que lleguen después de su instante de reproducción planificado se descartan y se consideran perdidos.

¿Qué valor sería adecuado para q? La telefonía por Internet puede permitir retardos de hasta unos 400 milisegundos, aunque se consigue una experiencia interactiva más satisfactoria con valores de q menores. Por otro lado, si hacemos q mucho menor que 400 milisegundos, entonces muchos paquetes podrían no llegar antes de su instante de reproducción planificado debido a la fluctuación de los paquetes por causa de la red. Por simplificar, digamos que si suelen experimentarse grandes variaciones en los retardos terminal a terminal es preferible utilizar un valor de q grande; por el contrario, si el retardo es pequeño y las variaciones del mismo también es preferible emplear un valor de q pequeño, tal vez inferior a 150 milisegundos.

En la Figura 7.5 se ilustra este compromiso entre el retardo de reproducción y la tasa de pérdida de paquetes. La figura muestra los instantes en que se generan los paquetes y los instantes en que esos paquetes se reproducen para un único periodo de conversación. En la figura se consideran dos retardos de reproducción iniciales diferentes. Como se muestra en la línea escalonada de la izquierda, el emisor genera paquetes a intervalos periódicos (por ejemplo, cada 20 milisegundos). El primer paquete de ese periodo de conversación se recibe

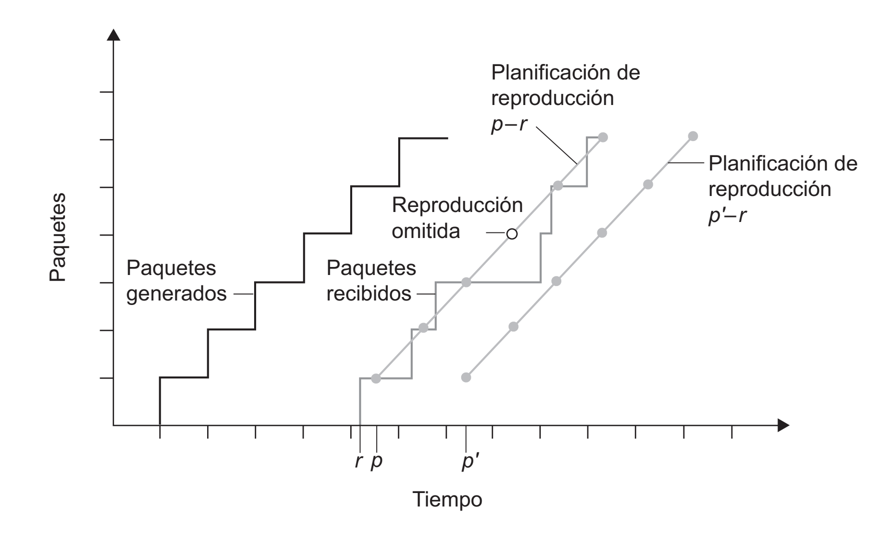


Figura 7.5 • Pérdidas de paquetes para distintos retardos de reproducción fijos.

en el instante r. Como se muestra en la figura, los instantes de llegada de los paquetes sucesivos no están espaciados de manera uniforme debido a las fluctuaciones de la red.

Para la primera planificación de reproducción se fija un retardo de reproducción inicial fijo de $p - r$. Con esta planificación el cuarto paquete no llega antes de su instante de reproducción planificado, por lo que el reproductor considera que se ha perdido. En la segunda planificación de reproducción el retardo de reproducción inicial fijo se hace igual a $p \; - r$. Para esta planificación todos los paquetes llegan antes de su instante de reproducción planificado, por lo que no se produce ninguna pérdida.

Retardo de reproducción adaptativo

El ejemplo anterior ilustra un importante compromiso entre el retardo y la tasa de pérdidas que surge siempre a la hora de diseñar una estrategia de reproducción con retardos de reproducción fijos. Si hacemos que el retardo inicial de reproducción sea grande, la mayoría de los paquetes llegarán a tiempo y por tanto habrá una tasa de pérdidas despreciable; sin embargo, para servicios interactivos como la telefonía por Internet, esos grandes retardos pueden resultar molestos, si no intolerables. Idealmente, lo que queremos es minimizar el retardo de reproducción bajo la restricción de que la tasa de pérdidas sea inferior a un cierto porcentaje precisado y pequeño.

La forma natural de tratar con este compromiso es estimar el retardo de la red y la varianza de ese retardo y ajustar el retardo de reproducción correspondientemente al principio de cada periodo de conversación. Este ajuste adaptativo de los retardos de reproducción al principio de los periodos de conversación hará que los periodos de silencio del emisor se compriman y estiren; sin embargo, la compresión y el alargamiento de esos silencios en una proporción razonable no resulta perceptible durante la conversación.

De acuerdo con lo expuesto en [Ramjee 1994], ahora vamos a describir un algoritmo genérico que puede utilizar el receptor para ajustar adaptativamente sus retardos de reproducción. Con este fin, sean

t_i = la marca de tiempo del paquete *i-ésimo*, es decir, el instante en que el paquete fue generado por el emisor

r_i = el instante en el que el paquete i llega al receptor

p_i = el instante en el que el receptor reproduce el paquete i

El retardo terminal a terminal de la red para el paquete *i-ésimo* es $r_i - t_i$. Debido a la fluctuación de la red este retardo variará de un paquete a otro. Sea d_i una estimación del retardo *promedio* de la red al recibirse el paquete *i-ésimo*. Esta estimación se calcula a partir de la marca de tiempo de la forma siguiente:

$$d_i = (1 - u)\, d_{i-1} + u\, (r_i - t_i)$$

donde u es una constante fija (por ejemplo, u = 0,01). Por tanto, d_i es una media móvil de los retardos de red observados $r_1 - t_1, \ldots, r_i - t_i$. La estimación asigna un mayor peso a los retardos de red más recientemente observados que a los del pasado más distante. Esta forma de estimación no debería resultarle extraña al lector; una idea similar se emplea para estimar los tiempos de ida y vuelta en TCP, como hemos visto en el Capítulo 3. Sea v_i una estimación de la desviación media del retardo con respecto al retardo promedio estimado. Esta estimación también se construye a partir de las marcas de tiempo:

$$v_i = (1 - u)\, v_{i-1} + u \mid r_i - t_i - d_i \mid$$

Las estimaciones d_i y v_i se calculan para cada paquete recibido, aunque sólo se utilizan para determinar el punto de reproducción del primer paquete de cada periodo de conversación.

Una vez que se han calculado estas estimaciones, el receptor emplea el siguiente algoritmo para la reproducción de los paquetes. Si el paquete *i* es el primero de un periodo de conversación, su instante de reproducción p_i se calcula como:

$$p_i = t_i + d_i + Kv_i$$

donde K es una constante positiva (por ejemplo, $K = 4$). El propósito del término Kv_i es establecer el instante de reproducción lo suficientemente lejos en el futuro como para que sólo una pequeña fracción de los paquetes correspondientes a ese periodo de conversación que vayan llegando se pierdan debido a una llegada tardía. El instante de reproducción para los siguientes paquetes de un periodo de conversación se calcula mediante un desplazamiento con respecto al instante en el que el primer paquete de ese periodo de conversación se reproduce. En particular, sea

$$q_i = p_i - t_i$$

la diferencia temporal entre el instante en que se generó el primer paquete del periodo de conversación y el instante en se reprodujo. Si el paquete *j* también pertenece al mismo periodo de conversación se reproducirá en el instante

$$p_j = t_j + q_i$$

El algoritmo recién descrito tiene mucho sentido, asumiendo que el receptor pueda determinar si un paquete es el primero de su correspondiente periodo de conversación. Si no hay pérdidas de paquetes, entonces el receptor podrá determinar si el paquete *i* es el primero de su periodo de conversación comparando la marca de tiempo del paquete *i-ésimo* con la del paquete $(i - 1)$-*ésimo*. De hecho, si $t_i - t_{i-1} > 20$ milisegundos, entonces el receptor sabe que el paquete *i-ésimo* está dando inicio a un nuevo paquete de conversación. Pero ahora suponga que existen pérdidas ocasionales de paquetes. En este caso, dos paquetes sucesivos recibidos en el destino pueden tener marcas de tiempo que difieren en más de 20 milisegundos, a pesar de que ambos paquetes pertenecen al mismo periodo de conversación. Y es por esta razón por la que los números de secuencia resultan particularmente útiles. El receptor puede emplear los números de secuencia para determinar si esa diferencia de más de 20 milisegundos en las marcas de tiempo se debe a un nuevo periodo de conversación o a la pérdida de paquetes.

Flujos de audio y vídeo almacenado

Concluyamos esta sección con algunas palabras acerca de los flujos de audio y vídeo almacenado. Normalmente, las aplicaciones de audio/vídeo almacenado también utilizan los números de secuencia, las marcas de tiempo y el retardo de reproducción para aliviar o incluso eliminar los efectos de fluctuación de la red. Sin embargo, existe una importante diferencia entre el audio/vídeo interactivo en tiempo real y los flujos de audio/vídeo almacenado. Específicamente, los flujos de audio/vídeo almacenado pueden tolerar retardos significativamente mayores. De hecho, cuando un usuario solicita un clip de audio/vídeo, le resulta perfectamente aceptable esperar cinco segundos o más antes de que la reproducción

comience. Y la mayoría de los usuarios pueden tolerar también retardos similares después de llevar a cabo operaciones interactivas tales como un salto temporal dentro del flujo multimedia. Esta mayor tolerancia al retardo proporciona al desarrollador de aplicaciones una mayor flexibilidad a la hora de diseñar aplicaciones para datos multimedia almacenados.

7.3.3 Recuperación frente a pérdidas de paquetes

Hemos explicado con un cierto grado de detalle cómo una aplicación de telefonía por Internet puede enfrentarse a la fluctuación de los paquetes. Ahora vamos a describir brevemente diversos esquemas que tratan de preservar una calidad de audio aceptable en presencia del fenómeno de pérdida de paquetes. Dichos esquemas se denominan **esquemas de recuperación frente a pérdidas**. Aquí, definimos la pérdida de paquetes en un sentido amplio: un paquete se pierde si nunca llega al receptor o si llega después de su instante de reproducción planificado. Nuestro ejemplo de telefonía por Internet nos servirá de nuevo de contexto para describir los esquemas de recuperación frente a pérdidas.

Como hemos mencionado al principio de esta sección, la retransmisión de los paquetes perdidos generalmente no resulta apropiada en una aplicación interactiva en tiempo real como la de telefonía por Internet. De hecho, la retransmisión de un paquete que no haya llegado antes de su instante de reproducción planificado no tiene ningún sentido. Retransmitir un paquete que ha desbordado la cola de un router no suele poder hacerse con la suficiente rapidez. Debido a estas consideraciones, las aplicaciones de telefonía por Internet utilizan a menudo algún tipo de esquema de anticipación de pérdidas. Dos tipos de esquemas de anticipación de pérdidas son la **corrección de errores hacia adelante (FEC, *Forward Error Correction*)** y el **intercalado**.

Corrección de errores hacia adelante (FEC)

La idea básica de la técnica FEC consiste en añadir información redundante al flujo original de paquetes. A cambio de incrementar ligeramente la velocidad de transmisión del flujo de audio, esa información redundante puede utilizarse para reconstruir aproximaciones o versiones exactas de algunos de los paquetes perdidos. Siguiendo lo expuesto en [Bolot 1996] y [Perkins 1998], vamos a esbozar dos mecanismos FEC sencillos. El primero de ellos envía un fragmento redundante codificado después de cada n fragmentos. El fragmento redundante se obtiene aplicando la operación OR exclusiva a los n fragmentos originales [Shacham 1990]. De esta manera, si algún paquete del grupo de los $n + 1$ paquetes se pierde el receptor puede reconstruir completamente el paquete perdido. Pero si se pierden dos o más paquetes de un grupo, el receptor no podrá reconstruirlos. Manteniendo pequeño el tamaño del grupo, $n + 1$, un gran porcentaje de los paquetes perdidos puede recuperarse siempre que la tasa de pérdidas no sea excesiva. Sin embargo, cuanto más pequeño sea el tamaño del grupo, mayor será el incremento relativo de la velocidad de transmisión del flujo de audio. En particular, la velocidad de transmisión se incrementa según un factor de $1/n$; por ejemplo, si $n = 3$ la velocidad de transmisión se incrementa en un 33 por ciento. Además, este esquema sencillo incrementa el retardo de reproducción, ya que el receptor tiene que esperar a recibir el grupo de paquetes completo antes de poder iniciar la reproducción. Para ver detalles más prácticos acerca de cómo funciona el mecanismo FEC en el transporte de datos multimedia, consulte [RFC 2733].

El segundo mecanismo FEC consiste en enviar un flujo de audio de menor resolución como información redundante. Por ejemplo, el emisor podría crear un flujo de audio nomi-

nal y otro flujo correspondiente de baja resolución y baja tasa de bit. (El flujo nominal podría ser una codificación PCM a 64 kbps y el flujo de menor calidad podría ser una codificación GSM a 13 kbps.) El flujo de baja velocidad de bit se denomina flujo redundante. Como se muestra en la Figura 7.6, el emisor construye el *n-ésimo* paquete tomando el *n-ésimo* fragmento del flujo nominal y añadiéndole el (*n* – 1)-*ésimo* fragmento del flujo redundante. De este manera, cuando existan pérdidas de paquetes no consecutivos, el receptor podrá ocultar esa pérdida reproduciendo el fragmento codificado de baja tasa de bit que llegue con el siguiente paquete. Por supuesto, los fragmentos de baja tasa de bit proporcionan una menor calidad que los fragmentos nominales. Sin embargo, un flujo compuesto por una gran mayoría de fragmentos de alta calidad, algunos fragmentos ocasionales de baja calidad y en el que no falte ningún fragmento proporciona una buena calidad global de audio. Observe que, en este esquema, el receptor sólo tiene que recibir dos paquetes para comenzar a reproducir, por lo que el incremento en el retardo de reproducción es pequeño. Además, si la codificación a baja velocidad es muy inferior a la codificación nominal, entonces el incremento en la velocidad de transmisión será pequeño.

Para poder resolver el problema de la pérdida de paquetes consecutivos podemos utilizar una sencilla variante. En lugar de añadir simplemente el (*n* – 1)-*ésimo* fragmento de tasa de bit baja al *n-ésimo* fragmento nominal, el emisor puede añadir el (*n* – 1)-*ésimo* y el (*n* – 2)-*ésimo* fragmentos de baja tasa de bit, o añadir el (*n* – 1)-*ésimo* y el (*n* – 3)-*ésimo* fragmentos de baja tasa de bit, etc. Añadiendo más fragmentos de baja tasa de bit al fragmento nominal, la calidad de audio en el receptor será aceptable para una mayor variedad de entornos de entrega de paquetes de mejor esfuerzo en presencia de pérdidas. Por otro lado, cada fragmento adicional incrementa el ancho de banda de transmisión y el retardo de reproducción.

RAT [RAT 2009] es una aplicación de telefonía por Internet bien documentada que utiliza mecanismos FEC. Puede transmitir flujos de audio de menor calidad junto con el flujo de audio nominal, tal como hemos descrito anteriormente. Consulte también [Rosenberg 2000].

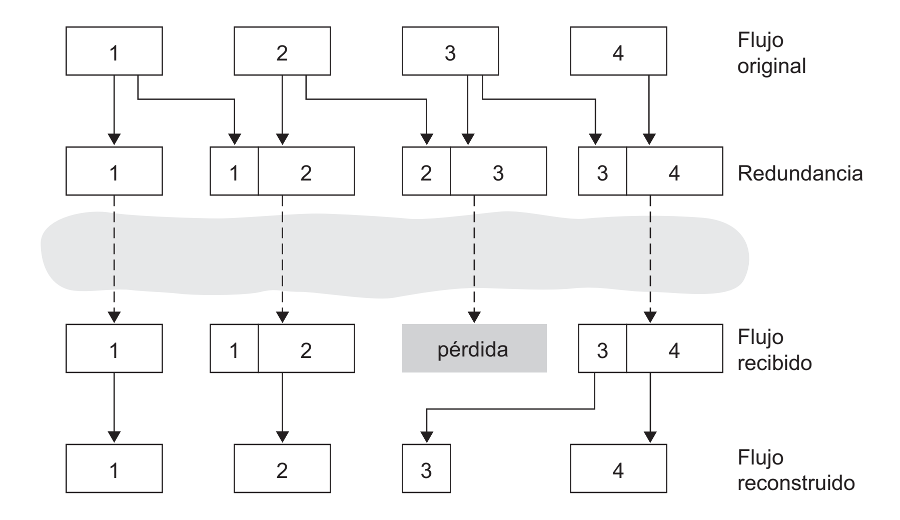


Figura 7.6 • Inserción de información redundante de menor calidad.

Intercalado

Como alternativa a la transmisión de información redundante, una aplicación de telefonía por Internet puede enviar el audio intercalado. Como se muestra en la Figura 7.7, el emisor reordena las unidades de datos de audio antes de su transmisión, de forma que las unidades originalmente adyacentes están separadas por un cierta distancia dentro del flujo transmitido. El intercalado puede mitigar el efecto de la pérdida de paquetes. Si, por ejemplo, las unidades tienen una longitud de 5 milisegundos y los fragmentos tienen una longitud de 20 milisegundos (es decir, hay cuatro unidades por fragmento), entonces el primer fragmento podría contener las unidades 1, 5, 9 y 13; el segundo fragmento podría contener las unidades 2, 6, 10 y 14, y así sucesivamente. La Figura 7.7 muestra que la pérdida de un único paquete en un flujo intercalado da como resultado una serie de múltiples huecos pequeños en el flujo reconstruido, en lugar del único hueco de gran tamaño que aparecería si tuviéramos un flujo no intercalado.

El intercalado puede mejorar significativamente la calidad percibida en un flujo de audio [Perkins 1998]. También tiene una baja sobrecarga de datos administrativos. La desventaja obvia del intercalado es que incrementa la latencia. Esto limita su uso en las aplicaciones interactivas como la telefonía por Internet, aunque puede funcionar bien para el envío de flujos de audio almacenado. Una de las principales ventajas del intercalado es que no incrementa los requisitos de ancho de banda de un flujo.

Reparación de los flujos de audio dañados en el receptor

Los esquemas de recuperación basados en el receptor tratan de generar un sustituto para un paquete perdido que sea similar al original. Como se explica en [Perkins 1998] es posible hacer esto, ya que las señales de audio, y en particular la voz, exhiben una gran cantidad de

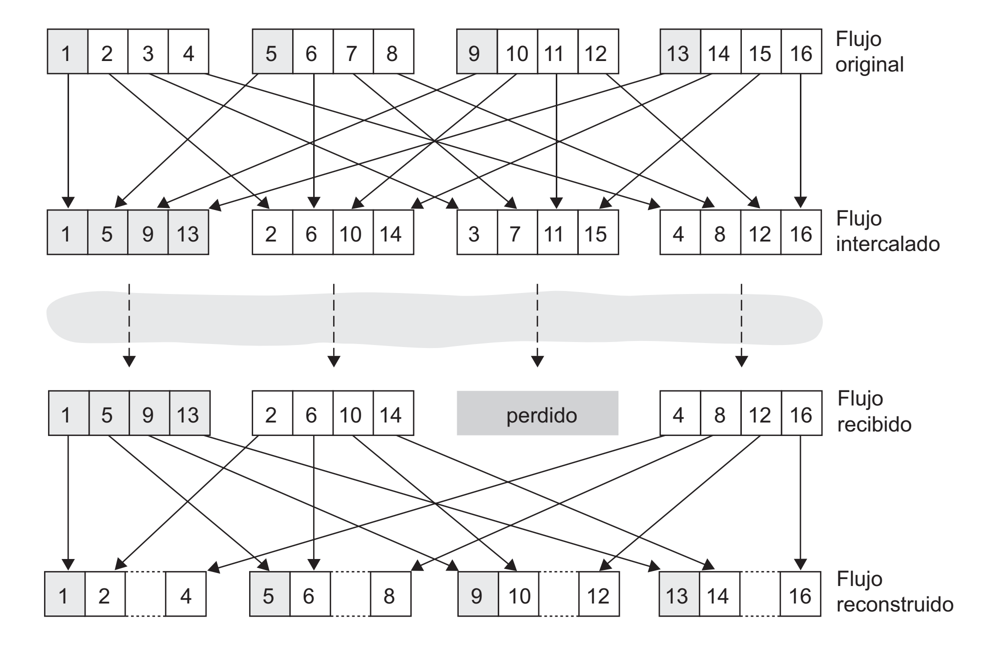


Figura 7.7 • Envío de audio intercalado.

auto-similitud a corto plazo. Por esto, estas técnicas funcionan para tasas de pérdida relativamente bajas (de menos del 15 por ciento) y para paquetes de pequeño tamaño (4-40 milisegundos). Cuando la longitud de la pérdida se aproxima a la longitud de un fonema (5-100 milisegundos) estas técnicas dejan de funcionar ya que el oyente podría perderse fonemas completos.

Quizá la forma más simple de recuperación por parte del receptor es la repetición de paquetes. La repetición de paquetes sustituye los paquetes perdidos con copias de los paquetes que hayan llegado inmediatamente antes de la pérdida. Esta técnica tiene una baja complejidad computacional y proporciona unos resultados razonablemente buenos. Otra forma de recuperación basada en el receptor es la interpolación, que utiliza el audio anterior y posterior a la pérdida para interpolar un paquete adecuado que permita cubrir el hueco. La interpolación funciona algo mejor que la repetición de paquetes, pero requiere un uso significativamente mayor de recursos de computación [Perkins 1998].

7.3.4 Distribución multimedia en la red Internet actual: redes de distribución de contenido

Dado que las velocidades de transmisión de flujos de vídeo varían desde centenares de kbps para los vídeos de baja resolución hasta varios Mbps para el vídeo de calidad DVD, la tarea de enviar flujos de vídeo almacenado bajo demanda a un gran número de usuarios distribuidos geográficamente parece plantear un desafío de gran magnitud. El enfoque más simple sería almacenar el vídeo en un único servidor y simplemente transmitir el flujo desde un servidor de vídeo (o granja de servidores) a un cliente para cada solicitud de cliente, como se explica en la Sección 7.2. Pero existen dos problemas obvios con esta solución. En primer lugar, puesto que un cliente puede encontrarse muy lejos del servidor, los paquetes de servidor a cliente pueden pasar a través de muchos ISP, incrementando la probabilidad de que los retardos y las pérdidas sean significativos. En segundo lugar, si el vídeo es muy popular probablemente haya que enviarlo múltiples veces a través de los mismos ISP (y a través de los mismos enlaces de comunicaciones), consumiendo así un ancho de banda significativo. En el Capítulo 2 hemos explicado cómo el almacenamiento en caché puede aliviar estos problemas. Aunque allí analizamos el almacenamiento en caché en términos de contenido web tradicional, debería estar claro que el almacenamiento en caché también es apropiado para contenido multimedia, como el audio y el vídeo almacenado. En esta sección vamos a hablar de las **redes de distribución de contenido (CDN, *Content Distribution Networks*)**, que proporcionan una solución alternativa para la distribución de contenido multimedia almacenado (así como para la distribución de contenido web tradicional).

Las redes CDN se basan en la filosofía de que si el cliente no puede acercarse al contenido (porque la ruta de entrega de mejor esfuerzo desde el servidor al cliente no puede soportar los flujos de vídeo), entonces el contenido debe ser llevado hasta el cliente. Las redes CDN utilizan, por tanto, un modelo diferente que las cachés web. Para una red CDN, los clientes de pago ya no son los ISP, sino los proveedores de contenido. Un proveedor de contenido que tenga un vídeo para distribuir (como por ejemplo CNN) paga a una compañía CDN (como Akamai) para que lleve su vídeo a los usuarios que lo soliciten con el menor retardo posible.

Una compañía CDN suele proporcionar su servicio de distribución de contenido de la forma siguiente:

1. La compañía CDN instala centenares de **servidores CDN** por todo Internet. Normalmente, la empresa CDN coloca los servidores CDN en **centros de datos**, que suelen estar ubicados en los ISP de nivel inferior, cerca de las redes de acceso a los ISP y de los clientes.
2. La compañía CDN replica el contenido de sus clientes en los servidores CDN. Cada vez que un cliente actualiza su contenido, la empresa CDN redistribuye el nuevo contenido a los servidores CDN.
3. La empresa CDN proporciona un mecanismo para que, cuando un cliente solicita el contenido, ese contenido sea proporcionado por el servidor CDN que mejor pueda entregárselo a ese cliente específico. Este servidor puede ser el servidor más próximo al cliente (quizá en el mismo ISP que el cliente) o ser un servidor CDN que disponga de una ruta libre de congestión hacia el cliente.

La Figura 7.8 muestra la interacción entre el proveedor de contenido y la empresa CDN. El proveedor de contenido determina primero cuál de sus objetos (por ejemplo, vídeos) quiere que la CDN distribuya. (El proveedor de contenido distribuye los restantes objetos sin que la CDN intervenga.) El proveedor de contenido etiqueta el contenido y lo envía a un nodo CDN, que a su vez replica el contenido y lo envía a una serie de servidores CDN seleccionados. La empresa CDN puede poseer una red privada para enviar el contenido desde el nodo CDN a los servidores CDN. Cada vez que el proveedor de contenido modifica un

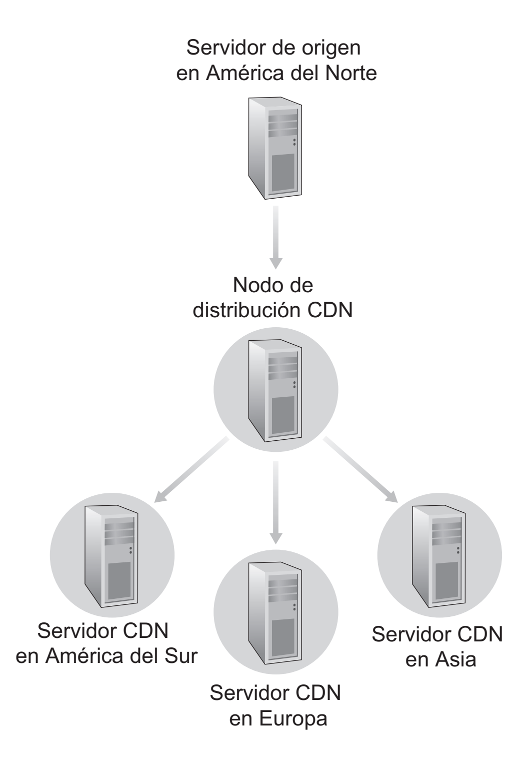


Figura 7.8 • La red CDN envía los objetos etiquetados por el proveedor de contenido a sus servidores CDN.

objeto distribuido CDN envía la versión nueva al nodo CDN, que de nuevo replica inmediatamente el objeto y los distribuye a los servidores CDN. Es importante recordar que cada servidor CDN normalmente contiene objetos de muchos proveedores de contenido.

Ahora viene la pregunta interesante. Cuando un navegador en el host de un usuario recibe la orden de extraer un objeto específico (identificado mediante un URL), ¿cómo determina el navegador si debe extraer el objeto del servidor origen o de uno de los servidores CDN? Normalmente, las redes CDN hacen uso de la técnica de redirección DNS para guiar a los navegadores hacia el servidor correcto [Kangasharju 2000].

Por ejemplo, suponga que el nombre de host del proveedor de contenido es `www.foo.com`. Suponga que el nombre de la empresa CDN es `cdn.com`. Suponga además que el proveedor de contenido sólo quiere que la CDN distribuya sus archivos de vídeo mpeg, mientras que todos los demás objetos, incluyendo las páginas HTML de base, serán distribuidas directamente por el proveedor de contenido. Para conseguir esto, el proveedor de contenido modifica todos los objetos HTML del servidor de origen de modo que todos los URL de los archivos de vídeo vayan precedidos por `http://www.cdn.com`. Por tanto, si un archivo HTML en el proveedor de contenido tenía originalmente una referencia a `http://www.foo.com/sports/highlights.mpg`, el proveedor de contenido etiquetaría este objeto sustituyendo la referencia en el archivo HTML por `http://www.cdn.com/www.foo.com/sports/highlights.mpg`.

Cuando un navegador solicita una página web que contiene el vídeo `highlights.mpg`, tendrán lugar las siguientes acciones:

1. El navegador envía la solicitud correspondiente al objeto HTML de base hacia el servidor origen, `www.foo.com`, el cual envía el objeto HTML solicitado al navegador. El navegador analiza el archivo HTML y encuentra la referencia a `http://www.cdn.com/www.foo.com/sports/highlights.mpg`.
2. El navegador hace entonces una búsqueda DNS de `www.cdn.com`, que es el nombre de host correspondiente al URL referenciado. El sistema DNS está configurado de forma que todas las consultas acerca de `www.cdn.com` que lleguen a un servidor DNS raíz se envíen a un servidor DNS autoritativo para `www.cdn.com`. Cuando el servidor DNS autoritativo recibe la consulta, extrae la dirección IP del navegador solicitante. Utilizando un mapa interno de la red que ha construido para toda Internet, el servidor DNS de la empresa CDN devuelve la dirección IP del servidor CDN que probablement sea el más adecuado para el navegador solicitante (a menudo será el servidor CDN más próximo al navegador).
3. El cliente DNS en el host que realiza la solicitud recibe una respuesta DNS con la dirección IP. El navegador envía entonces su solicitud HTTP al servidor CDN que tiene esa dirección IP. El navegador obtiene el archivo de vídeo `highlights.mpg` de ese servidor CDN. Para subsiguientes solicitudes dirigidas a `www.cdn.com`, el cliente continuará utilizando el mismo servidor CDN, ya que la dirección IP de `www.cdn.com` se encontrará en la caché DNS (en el host cliente o en el servidor de nombres DNS local).

En resumen, como se muestra en la Figura 7.9, el host que realiza la solicitud acude primero al servidor web origen para obtener el objeto HTML de base y luego al servidor DNS autoritativo correspondiente a la empresa CDN para obtener la dirección IP del mejor servidor CDN. Por último, acude a ese servidor CDN para obtener el vídeo. Observe que no hace falta realizar ningún cambio en HTTP, DNS o en el navegador para implementar este esquema de distribución.

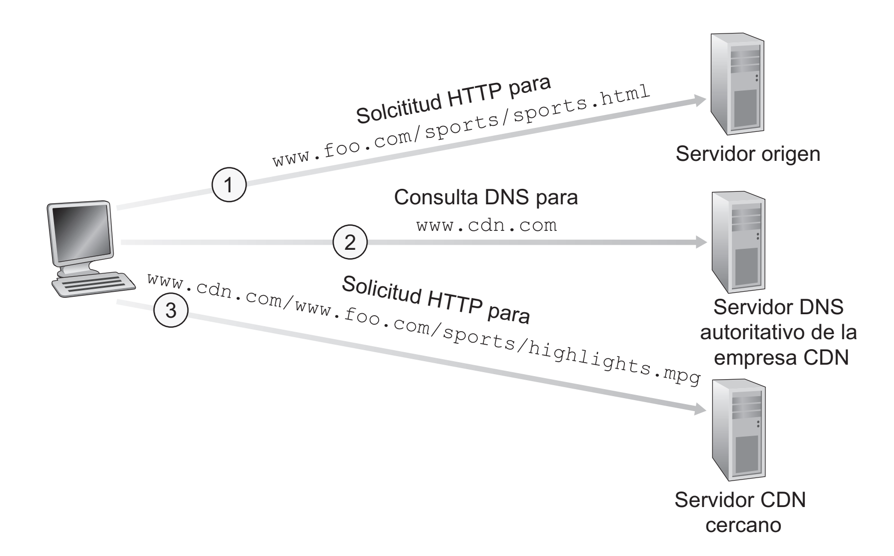


Figura 7.9 • Las redes CDN utilizan el sistema DNS para dirigir las solicitudes a un servidor CDN cercano.

Lo que nos queda por explicar es cómo una empresa CDN determina el "mejor" servidor CDN para el host solicitante. Aunque cada empresa CDN tiene su propia manera propietaria de hacer esto, no es difícil hacerse una idea aproximada de qué es lo que hacen. Para cada ISP de acceso a Internet (que contiene potenciales clientes que realizan solicitudes), la empresa CDN tiene anotado cuál es el mejor servidor CDN. La empresa CDN determina cuál es el mejor servidor CDN basándose en su conocimiento de las tablas de enrutamiento Internet (específicamente de las tablas BGP de las que hemos hablado en el Capítulo 4), en las estimaciones del tiempo de ida y vuelta y en otras medidas de las que dispone para las comunicaciones entre sus diversos servidores en las distintas redes de acceso; consulte [Verma 2001] para ver un análisis más detallado. De este forma, la empresa CDN estima qué servidor CDN proporcionará el mejor servicio de entrega de mejor esfuerzo a cada ISP. La empresa CDN hace esto para un gran número de ISP de acceso en Internet y emplea dicha información para configurar el servidor DNS autoritativo. Para ver un análisis del envío de flujos multimedia desde el punto de vista de una gran empresa CDN operativa (Akamai), consulte [Sripanidkulchpai 2004]. Para ver los recientes desarrollos en la investigación de las redes CDN, consulte [Krishnamurthy 2001; Mao 2002; Saroiu 2002; Freedman 2004; Su 2006; Huang 2008].

7.3.5 Dimensionamiento de las redes con servicio de entrega de mejor esfuerzo para proporcionar calidad de servicio

En las secciones anteriores hemos visto cómo diversas técnicas del nivel de aplicación, como la reproducción de paquetes, los mecanismos FEC, el intercalado de paquetes en los hosts y una infraestructura CDN implantada por toda la red pueden mejorar la calidad de las aplicaciones multimedia en la red Internet actual que dispone de un servicio de entrega de mejor esfuerzo. Fundamentalmente, las dificultades para dar soporte a las aplicaciones multimedia surgen de sus estrictos requisitos de rendimiento (bajo retardo de paquetes terminal

a terminal, baja fluctuación del retardo y baja tasa de pérdidas) y del hecho de que el retardo de los paquetes, la fluctuación de los retardos y las pérdidas se producen cuando la red está congestionada. Una técnica definitiva para mejorar la calidad de las aplicaciones multimedia (una técnica que a menudo puede emplearse para resolver prácticamente cualquier problema en el que los recursos estén restringidos) es simplemente "meter dinero" y evitar desde el principio que exista una contienda por los recursos. En el caso de las aplicaciones multimedia en red, esto quiere decir proporcionar suficiente capacidad de enlace por toda la red, de modo que nunca se produzca (o sólo se produzca muy raramente) una congestión en la red, con los consiguientes retardos y pérdidas de paquetes. Con suficiente capacidad de enlace, los paquetes podrían atravesar la red Internet de hoy día sin retardos de puesta en cola y sin pérdidas. Desde muchos puntos de vista, ésta es una situación ideal: las aplicaciones multimedia funcionarían perfectamente, los usuarios estarían contentos y todo esto podría conseguirse sin efectuar ningún cambio en la arquitectura Internet basada en un servicio de entrega de mejor esfuerzo. Por supuesto, la cuestión es cuánta capacidad es "suficiente" para conseguir esta situación paradisiaca y si los costes de proporcionar un ancho de banda "suficiente" resultan prácticos desde el punto de vista comercial para los ISP.

La cuestión de cuánta capacidad proporcionar en los enlaces de la red para una cierta topología dada, con el fin de conseguir un determinado nivel de rendimiento terminal a terminal, se suele denominar **aprovisionamiento de ancho de banda**. El problema todavía más complicado de cómo diseñar una topología de red (dónde colocar los routers, cómo interconectar los routers mediante enlaces y qué capacidad asignar a los enlaces) para conseguir un nivel prefijado de rendimiento terminal a terminal es un problema de diseño de redes que a menudo de denomina **dimensionamiento de la red**. Tanto el aprovisionamiento de ancho de banda como el dimensionamiento de la red son temas complejos que caen fuera del alcance de este libro. Sin embargo, conviene decir que es necesario resolver los siguientes problemas para poder predecir el rendimiento de nivel de aplicación entre dos puntos terminales de la red y así poder prever una capacidad suficiente como para satisfacer los requisitos de rendimiento de una aplicación.

- *Modelos de demanda de tráfico entre puntos terminales de la red.* Los modelos pueden tener que especificarse tanto en el nivel de llamada (por ejemplo, usuarios "que llegan" a la red e inician aplicaciones terminal a terminal) como en el nivel de paquetes (por ejemplo, número de paquetes generados por las aplicaciones activas). Observe que la carga de trabajo puede variar a lo largo del tiempo.
- *Requisitos de rendimiento bien definidos.* Por ejemplo, un requisito de rendimiento para el soporte de tráfico sensible al retardo, como el de las aplicaciones interactivas de audio/vídeo, puede ser que la probabilidad de que el retardo terminal a terminal de la aplicación supere un determinado umbral máximo tolerable sea inferior a un cierto valor pequeño [Fraleigh 2003].
- *Modelos para predecir el rendimiento terminal a terminal para un modelo de carga de trabajo determinado, junto con técnicas para encontrar una asignación de coste mínimo del ancho de banda que permita satisfacer los requisitos de todos los usuarios.* En este aspecto, los investigadores están trabajando arduamente para desarrollar modelos (véase la Sección 1.4) que permitan cuantificar el rendimiento para una carga de trabajo determinada, junto con técnicas de optimización para hallar las asignaciones de ancho de banda de coste mínimo que satisfagan los requisitos de rendimiento.

Dado que la red Internet actual basada en un servicio de entrega de mejor esfuerzo podría (desde el punto de vista tecnológico) soportar tráfico multimedia con un rendimiento apropiado si estuviera dimensionada para hacerlo, la pregunta natural es por qué la red Internet de hoy día no lo hace. Las respuestas son principalmente económicas y organizativas. Desde el punto de vista económico, ¿estarían dispuestos los usuarios a pagar a sus ISP el suficiente dinero como para que los ISP instalaran un ancho de banda suficiente para soportar aplicaciones multimedia sobre la actual Internet? Las cuestiones organizativas son quizá aún más difíciles. Observe que una ruta terminal a terminal entre dos puntos extremos de una comunicación multimedia tendrá que pasar por las redes de múltiples ISP. Desde el punto de vista organizativo, ¿estarían dispuestos estos ISP a cooperar (tal vez compartiendo los recursos) para asegurar que la ruta terminal a terminal esté adecuadamente dimensionada y dé soporte a las aplicaciones multimedia? Para ver un análisis desde la perspectiva de estos problemas económicos y organizativos, consulte [Davies 2005]. Para ver un análisis del dimensionamiento de redes troncales de nivel 1 con el fin de dar soporte a tráfico sensible al retardo, consulte [Fraleigh 2003].

7.4 Protocolos para aplicaciones interactivas en tiempo real

Las aplicaciones interactivas en tiempo real, incluidas la telefonía por Internet y la videoconferencia, prometen controlar gran parte del crecimiento futuro de Internet. Por tanto, no debe sorprendernos que organismos de estandarización como IETF e ITU hayan estado ocupados durante muchos años (¡y continúan estándolo!) en el establecimiento de estándares para esta clase de aplicaciones. Disponiendo de los estándares apropiados para las aplicaciones interactivas en tiempo real, empresas independientes podrán crear nuevos y convincentes productos que interoperen entre sí. En esta sección vamos a examinar RTP, SIP y H.323 para aplicaciones interactivas en tiempo real. Estos tres conjuntos de estándares disfrutan de una amplia implementación en los productos industriales.

7.4.1 RTP

En la sección anterior hemos visto que el lado emisor de una aplicación multimedia añade campos de cabecera a los fragmentos de audio/vídeo antes de pasarlos a la capa de transporte. Estos campos de cabecera incluyen números de secuencia y marcas de tiempo. Dado que la mayoría de las aplicaciones multimedia de red pueden hacer uso de los números de secuencia y de las marcas de tiempo, es conveniente disponer de una estructura de paquete estandarizada que incluya campos para los datos de audio/vídeo, los números de secuencia y las marcas de tiempo, así como otros campos potencialmente útiles. RTP, definido en el documento RFC 3550, es este estándar. RTP puede emplearse para transportar formatos comunes como PCM, GSM y MP3 para sonido y MPEG y H.263 para vídeo. También se puede utilizar para transportar formatos de sonido y vídeo propietarios. Actualmente RTP disfruta de una amplia implementación en centenares de productos y prototipos de investigación. Además es complementario de otros importantes protocolos interactivos de tiempo real, como SIP y H.323.

En esta sección proporcionamos una introducción a RTP y a su protocolo compañero RTCP. Animamos a los lectores a visitar el sitio de RTP de Henning Schulzrinne [Schulzrinne-RTP 2009], que proporciona abundante información sobre el tema. También pueden visitar el sitio de RAT [RAT 2009], que documenta una aplicación de telefonía por Internet que emplea RTP.

Fundamentos de RTP

Normalmente, RTP se ejecuta sobre UDP. El lado emisor encapsula un fragmento multimedia dentro de un paquete RTP, luego encapsula ese paquete en un segmento UDP y después pasa el segmento a IP. El lado receptor extrae el paquete RTP del segmento UDP, a continuación extrae el fragmento multimedia del paquete RTP y lo pasa al reproductor multimedia para su decodificación y procesamiento.

Por ejemplo, considere el uso de RTP para transportar voz. Suponga que el origen de voz está codificado en PCM (es decir, la voz está muestreada, cuantizada y digitalizada) a 64 kbps. Suponga también que la aplicación recopila los datos codificados en fragmentos de 20 milisegundos; es decir, un fragmento tiene 160 bytes. El lado emisor precede a cada fragmento de datos de audio con una **cabecera RTP** que incluye el tipo de codificación audio, un número de secuencia y una marca de tiempo. La cabecera RTP normalmente tiene 12 bytes. El fragmento de audio junto con la cabecera RTP forman el **paquete RTP**. El paquete RTP se envía entonces a la interfaz de socket UDP. En el lado receptor, la aplicación recibe el paquete RTP procedente de su interfaz de socket. La aplicación extrae el fragmento de audio del paquete RTP y utiliza los campos de cabecera del mismo para decodificar y reproducir apropiadamente el fragmento.

Si una aplicación incorpora RTP, en lugar de un esquema propietario para especificar el tipo de carga útil, los números de secuencia o las marcas de tiempo, entonces la aplicación interoperará más fácilmente con las demás aplicaciones multimedia de red. Por ejemplo, si dos empresas distintas desarrollan software para telefonía por Internet y ambas incorporan en su producto el protocolo RTP, existirá la posibilidad de que un usuario que emplee uno de esos productos de telefonía por Internet pueda comunicarse con otro usuario que use el producto de la otra empresa. En la Sección 7.4.3 veremos que RTP suele utilizarse junto con los estándares de telefonía Internet.

Debemos destacar que RTP no proporciona ningún mecanismo para garantizar la entrega a tiempo de los datos ni ninguna otra garantía de calidad del servicio (QoS, *Quality-of-Service*); ni siquiera garantiza la entrega de paquetes o evita la entrega de paquetes desordenados. De hecho, la encapsulación RTP sólo se percibe en los sistemas terminales. Los routers no diferencian entre los datagramas IP que transportan paquetes RTP y los que no.

RTP permite que a cada origen (por ejemplo, una cámara o un micrófono) se le asigne su propio flujo independiente de paquetes RTP. Por ejemplo, para una videoconferencia entre dos participantes podrían abrirse cuatro flujos RTP: dos flujos para transmitir el audio (uno en cada dirección) y dos flujos para transmitir el vídeo (también uno en cada dirección). Sin embargo, muchas técnicas de codificación populares (entre las que se incluyen MPEG 1 y MPEG 2) empaquetan el audio y el vídeo en un mismo flujo durante el proceso de codificación. Cuando el codificador empaqueta el audio y el vídeo, entonces sólo se genera un flujo RTP en cada dirección.

Los paquetes RTP no están limitados a las aplicaciones de unidifusión. También pueden enviarse a través de árboles de multidifusión uno-a-muchos y muchos-a-muchos. En una

Tipo de carga útil	Número de secuencia	Marca de tiempo	Identificador de origen de sincronización	Diversos campos

Figura 7.10 • Campos de la cabecera RTP.

sesión de multidifusión muchos-a-muchos, normalmente todos los emisores y orígenes de la sesión utilizan el mismo grupo de multidifusión para enviar sus flujos RTP. Estos flujos multidifusión RTP conjuntos pertenecen a una **sesión RTP**, del mismo modo que los flujos de audio y de vídeo procedentes de múltiples emisores en una aplicación de videoconferencia.

Campos de cabecera de los paquetes RTP

Como se muestra en la Figura 7.10, los cuatro campos principales de la cabecera RTP son el tipo de carga útil, el número de secuencia, la marca de tiempo y el identificador de origen.

El campo Tipo de carga útil del paquete RTP tiene una longitud de 7 bits. En un flujo de audio, el campo Tipo de carga útil se emplea para indicar el tipo de codificación de audio (por ejemplo, PCM, modulación delta adaptativa, codificación predictiva lineal) que se está utilizando. Si un emisor decide cambiar el tipo de codificación en mitad de una sesión, puede informar al receptor de dicho cambio a través de este campo que define el tipo de carga útil. El emisor puede desear cambiar la codificación con el fin de aumentar la calidad del audio o para disminuir la tasa de bit del flujo RTP. En la Tabla 7.2 se enumeran algunos de los tipos de carga útil de audio a los que actualmente da soporte RTP.

Para un flujo de vídeo, el tipo de carga útil se utiliza para indicar el tipo de codificación de vídeo (por ejemplo, JPEG con movimiento, MPEG 1, MPEG 2, H.261). De nuevo, el emisor puede cambiar el tipo de codificación de vídeo sobre la marcha durante una sesión. En la Tabla 7.3 se enumeran algunos de los tipos de carga útil de vídeo soportados actualmente por RTP. Los restantes campos importantes son los siguientes:

- *Campo de número de secuencia.* El campo de número de secuencia tiene una longitud de 16 bits. El número de secuencia aumenta en una unidad para cada paquete RTP enviado

Número de tipo de carga útil	Formato de audio	Frecuencia de muestreo	Velocidad
0	PCM μ-law	8 kHz	64 kbps
1	1016	8 kHz	4,8 kbps
3	GSM	8 kHz	13 kbps
7	LPC	8 kHz	2,4 kbps
9	G.722	16 kHz	48–64 kbps
14	MPEG Audio	90 kHz	—
15	G.728	8 kHz	16 kbps

Tabla 7.2 • Tipos de carga útil de audio soportados por RTP.

Número de tipo de carga útil	Formato de vídeo
26	JPEG con movimiento
31	H.261
32	MPEG 1 vídeo
33	MPEG 2 vídeo

Tabla 7.3 • Algunos tipos de carga útil de vídeo soportados por RTP.

y puede ser utilizado por el receptor para detectar pérdidas de paquetes y restaurar la secuencia de paquetes. Por ejemplo, si el lado receptor de la aplicación recibe un flujo de paquetes RTP con un hueco entre los números de secuencia 86 y 89, entonces el receptor sabe que faltan los paquetes 87 y 88. El receptor puede entonces intentar ocultar los datos perdidos.

- *Campo de marca de tiempo.* El campo de marca de tiempo tiene una longitud de 32 bits y designa el instante de muestreo del primer byte del paquete de datos RTP. Como hemos visto en la sección anterior, el receptor puede utilizar marcas de tiempo para eliminar la fluctuación de los paquetes introducida por la red y para proporcionar una reproducción síncrona en el receptor. La marca de tiempo se obtiene de una señal de reloj de muestreo del emisor. Por ejemplo, para audio, la señal de reloj de marca de tiempo se incrementa en una unidad para cada periodo de muestreo (por ejemplo, cada 125 microsegundos para una señal de reloj de muestro de 8 kHz); si la aplicación de audio genera fragmentos que constan de 160 muestras codificadas, entonces la marca de tiempo se incrementa en 160 para cada paquete RTP cuando el origen está activo. La señal de reloj de marca de tiempo continúa aumentando a una velocidad constante incluso aunque el origen esté inactivo.
- *Identificador del origen de sincronización (SSRC, Synchronization Source Identifier).* El campo SSRC tiene una longitud de 32 bits. Este campo identifica el origen del flujo RTP. Normalmente, cada flujo de una sesión RTP tiene un SSRC distinto. El SSRC no es la dirección IP del emisor, sino un número que el origen asigna aleatoriamente cuando se inicia un nuevo flujo. La probabilidad de que dos flujos obtengan el mismo SSRC es muy pequeña. En el caso de que esto ocurriera, los dos orígenes deberán elegir un nuevo valor de SSRC.

Desarrollo de aplicaciones software con RTP

Existen dos enfoques para el desarrollo de una aplicación de red basada en RTP. El primer enfoque consiste en que el desarrollador de la aplicación incorpore manualmente RTP; es decir, que escriba realmente el código que lleve a cabo la encapsulación RTP en el lado emisor y el procesamiento RTP en el lado receptor. El segundo enfoque consiste en que el desarrollador de la aplicación utilice las bibliotecas RTP existentes (para los programadores de C) y las clases Java (para los programadores de Java) que realizan la encapsulación y el procesamiento para la aplicación. Puesto que es posible que esté interesado en escribir su primera aplicación multimedia de red utilizando RTP, a continuación vamos a elaborar un poco más cada uno de estos dos enfoques. (La tarea de programación disponible al final del capítulo le guiará a través de la creación de una aplicación RTP.) Haremos esto en el contexto de una comunicación de unidifusión (en lugar de en un contexto de multidifusión).

Recuerde del Capítulo 2 que la API de UDP requiere que el proceso emisor establezca, para cada segmento UDP que envía, la dirección de destino IP y el número de puerto de destino antes de enviar el paquete al socket UDP. A continuación el segmento UDP se transmitirá a través de Internet y (si el segmento no se pierde, por ejemplo, a causa de un desbordamiento del buffer del router) terminará llegando a la puerta del proceso receptor de la aplicación. Esta puerta queda completamente definida por la dirección IP de destino y el número de puerto de destino. De hecho, cualquier datagrama IP que contenga esta dirección IP de destino y ese número de puerto de destino será dirigido a la puerta UDP del proceso receptor. (La API de UDP también permite al desarrollador de la aplicación establecer el número de puerto de origen de UDP; sin embargo, este valor no tiene ningún efecto sobre el proceso al que se envía el segmento.) Es importante destacar que RTP no obliga a emplear un número de puerto específico. Cuando un desarrollador de aplicaciones crea una aplicación RTP, especifica los números de puerto para los dos lados de la aplicación.

Como parte de la tarea de programación de este capítulo, tendrá que escribir un servidor RTP que encapsule tramas de vídeo almacenado dentro de los paquetes RTP. Tendrá que hacerlo de forma manual; es decir, su aplicación tomará una trama de vídeo, añadirá las cabeceras RTP a la trama para crear un paquete RTP y luego pasará esa trama RTP al socket UDP. Para hacer esto necesitará crear campos contenedor para las diversas cabeceras RTP, incluyendo un campo de número de secuencia y un campo de marca de tiempo. Y para cada uno de los paquetes RTP creados, tendrá que definir apropiadamente el número de secuencia y la marca de tiempo. Tendrá que escribir explícitamente el código para llevar a cabo todas estas operaciones RTP en el lado emisor de su aplicación. Como se muestra en la Figura 7.11, su API para la red será la API estándar del socket UDP.

Un método alternativo (que no se realiza en la tarea de programación) consiste en utilizar una clase RTP en Java (o una biblioteca RTP en C para los programadores de C) para implementar las operaciones RTP. Como se muestra en la Figura 7.12, con este método el desarrollador de la aplicación puede tener la impresión de que RTP forma parte de la capa de transporte, con una API de RTP/UDP entre la capa de aplicación y la capa de transporte. Sin entrar en demasiados detalles (como los relativos a la clase/biblioteca), cuando se envía un fragmento de datos multimedia a la API, el lado emisor de la aplicación tiene que proporcionar a la interfaz el propio fragmento, un número de tipo de carga útil, un identificador SSRC y una marca de tiempo, además de un número de puerto de destino y una dirección IP de destino. Sólo comentar que Java Media Framework (JMF) incluye una implementación completa de RTP.

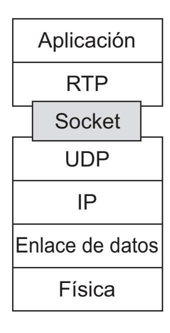


Figura 7.11 • RTP es parte de la aplicación y queda por encima del socket UDP.

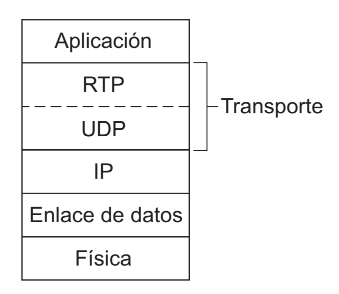


Figura 7.12 • RTP puede interpretarse como una subcapa de la capa de transporte.

7.4.2 Protocolo de control de RTP (RTCP)

El documento RFC 3550 también especifica el protocolo RTCP, un protocolo que una aplicación multimedia de red puede emplear junto con RTP. Como se muestra en el escenario de multidifusión de la Figura 7.13, cada uno de los participantes en una sesión RTP transmite paquetes RTCP a todos los demás participantes de la sesión mediante multidifusión IP. Normalmente, en una sesión RTP existe una única dirección de multidifusión y todos los paquetes RTP y RTCP pertenecientes a la sesión utilizan esa dirección de multidifusión. Los paquetes RTP y RTCP se distinguen entre sí por medio del uso de distintos números de puerto. (El número de puerto RTCP se establece para que sea igual al número de puerto RTP más uno.)

Los paquetes RTCP no encapsulan fragmentos de audio ni de vídeo. En lugar de ello, estos paquetes son enviados periódicamente y contienen informes del emisor y/o receptor que anuncian estadísticas que pueden ser útiles para la aplicación. Estas estadísticas incluyen el número de paquetes enviados, el número de paquetes perdidos y la fluctuación entre

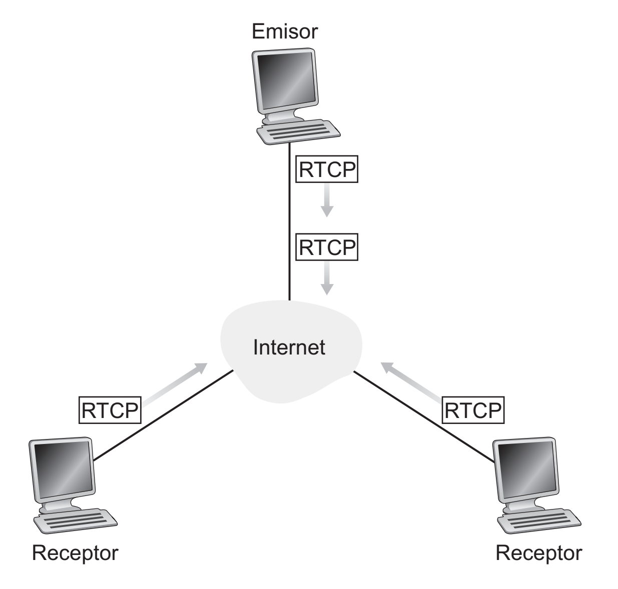


Figura 7.13 • Tanto los emisores como los receptores envían mensajes RTCP.

llegadas. La especificación de RTP [RFC 3550] no establece qué debe hacer la aplicación con esta información de realimentación, por lo que esta cuestión depende del desarrollador de la aplicación. Por ejemplo, los emisores pueden utilizar la información de realimentación para modificar sus velocidades de transmisión, o también se puede emplear con propósitos de diagnóstico; por ejemplo, los receptores pueden determinar si existen problemas de carácter local, regional o global.

Tipos de paquetes RTCP

Para cada flujo RTP que recibe un receptor como parte de una sesión, el receptor genera un informe de recepción. El receptor agrega sus informes de recepción en un único paquete RTCP, el cual es enviado al árbol de multidifusión que conecta a todos los participantes de la sesión. El informe de recepción incluye varios campos, siendo los más importantes los siguientes:

- El identificador SSRC del flujo RTP para el que se ha generado el informe de recepción.
- La fracción de paquetes perdidos dentro del flujo RTP. Cada receptor calcula el número de paquetes RTP perdidos dividido entre el número de paquetes RTP enviados como parte del flujo. Si un emisor recibe informes de recepción que indican que a los receptores sólo está llegando una pequeña fracción de los paquetes transmitidos por el emisor puede cambiar a una tasa de codificación menor, con el fin de disminuir la congestión de la red y mejorar la tasa de recepción.
- El último número de secuencia recibido en el flujo de paquetes RTP.
- La fluctuación entre llegadas, que es una estimación suavizada de la variación del tiempo que transcurre entre la llegada de paquetes sucesivos del flujo RTP.

El emisor crea y transmite paquetes de informe de emisor TCP para cada flujo RTP que transmite. Estos paquetes incluyen información acerca del flujo RTP, entre la que se incluye:

- El identificador SSRC del flujo RTP.
- La marca de tiempo y el tiempo absoluto del paquete RTP más recientemente generado en el flujo.
- El número de paquetes enviados en el flujo.
- El número de bytes enviados en el flujo.

Los informes del emisor se pueden utilizar para sincronizar los distintos flujos multimedia dentro de una sesión RTP. Por ejemplo, consideremos una aplicación de videoconferencia en la que cada emisor genera dos flujos RTP independientes, uno para el vídeo y otro para el audio. Las marcas de tiempo de estos paquetes RTP están ligadas a los relojes de muestreo del vídeo y del audio y no están relacionadas con el *tiempo absoluto* (es decir, el tiempo real). Cada informe de emisor RTCP contiene, para el paquete generado más recientemente en el flujo RTP asociado, la marca de tiempo del paquete RTP y el tiempo absoluto de cuando el paquete fue creado. Por tanto, los paquetes de informe de emisor RTCP asocian el reloj de muestreo con el reloj de tiempo real. Los receptores pueden utilizar esta asociación de los informes de emisor RTCP para sincronizar la reproducción del audio y del vídeo.

Para cada flujo RTP que transmite un emisor, éste también crea y transmite paquetes de descripción del origen. Estos paquetes contienen información acerca del origen, como la

dirección de correo electrónico del emisor, el nombre del mismo y la aplicación que genera el flujo RTP. También incluye el SSRC del flujo RTP asociado. Estos paquetes proporcionan una correspondencia entre el identificador del origen (es decir, el SSRC) y el nombre del usuario/host.

Los paquetes RTCP son apilables; es decir, los informes de recepción del receptor, los informes del emisor y los descriptores del origen se pueden concatenar en un mismo paquete. El paquete resultante se encapsula entonces en un segmento UDP que se reenvía sobre el árbol de multidifusión.

Escalado del ancho de banda RTCP

Es posible que haya observado que RTCP tiene un potencial problema de escalado. Por ejemplo, considere una sesión RTP que consta de un emisor y de un gran número de receptores. Si cada uno de los receptores genera periódicamente paquetes RTCP, entonces la velocidad agregada de transmisión de los paquetes RTCP puede exceder en gran medida la velocidad de los paquetes RTP enviados por el emisor. Observe que la cantidad de tráfico RTP enviada al árbol de multidifusión no cambia cuando aumenta el número de receptores, mientras que la cantidad de tráfico RTCP crece linealmente con el número de receptores. Para resolver este problema de escalado, RTCP modifica la velocidad a la que un participante envía paquetes RTCP sobre el árbol de multidifusión como una función del número de participantes en la sesión. Además, dado que cada participante envía paquetes de control a todos los demás, cada participante podrá estimar el número total de participantes en la sesión [Friedman 1999].

RTCP intenta limitar su tráfico a un 5 por ciento del ancho de banda de la sesión. Por ejemplo, suponga que hay un emisor que está enviando vídeo a una velocidad de 2 Mbps. En este caso, RTCP intenta limitar su tráfico al 5 por ciento de 2 Mbps, es decir, a 100 kbps, de la manera siguiente: el protocolo proporciona el 75 por ciento de esta velocidad, es decir, 75 kbps, a los receptores; y el restante 25 por ciento de la velocidad, 25 kbps, al emisor. Los 75 kbps dedicados a los receptores son compartidos de forma equitativa entre los receptores. Por tanto, si hay R receptores, entonces cada uno de ellos puede enviar tráfico RTCP a una velocidad de $75/R$ kbps, y el emisor podrá enviar tráfico RTCP a una velocidad de 25 kbps. Un participante (un emisor o un receptor) determina el periodo de transmisión de paquetes RTCP de forma dinámica calculando el tamaño promedio de paquete RTCP (a lo largo de la sesión completa) y dividiendo dicho valor entre su velocidad asignada. En resumen, para un emisor, el periodo de transmisión de los paquetes RTCP es:

$$T = \frac{\text{Número de emisores}}{0{,}25 \cdot 0{,}05 \cdot \text{ancho de banda de la sesión}} \text{ (tamaño medio de los paquetes RTCP)}$$

Y el periodo de transmisión de los paquetes RTCP para un receptor es:

$$T = \frac{\text{Número de receptores}}{0{,}75 \quad 0{,}05 \quad \text{ancho de banda de la sesión}} \text{ (tamaño medio de los paquetes RTCP)}$$

7.4.3 SIP

Imagine un mundo en el que, cuando se encuentra trabajando en su PC, sus llamadas de teléfono llegan a su PC a través de Internet. Cuando se levanta y camina, sus llamadas telefóni-

cas son enrutadas automáticamente a su PDA. Y cuando está conduciendo, sus llamadas son automáticamente enrutadas a algún dispositivo Internet disponible en su coche. En ese mismo mundo, también es posible estar participando en una conferencia telefónica y acceder a un libro de direcciones para llamar e invitar a otros para que participen en la conferencia. Los demás participantes pueden encontrarse sentados frente a sus equipos PC, paseando con sus PDA o conduciendo sus automóviles (independientemente de dónde estén, su invitación será enrutada de forma transparente). En ese mismo mundo, al navegar por la página principal de un individuo existirá un enlace "Llámame" en el que al hacer clic se establecerá una sesión de telefonía por Internet entre su PC y el propietario de la página (se encuentre donde se encuentre esa persona).

En ese mundo ya no existe una red de telefonía de conmutación de circuitos; en lugar de ello, todas las llamadas pasan a través de Internet de extremo a extremo. En este mundo, las empresas tampoco utilizan ya las PBX, es decir, circuitos conmutados locales que permiten gestionar las llamadas telefónicas entre empresas. En su lugar, el tráfico telefónico entre compañías fluye a través de la red LAN de alta velocidad de las mismas.

Todo esto puede sonarle a ciencia ficción y, por supuesto, las redes de comutación de circuitos y las PBX actuales no van a desaparecer completamente en un futuro cercano [Jiang 2001]. No obstante, existen protocolos y productos que pueden convertir esta visión en realidad. Entre los protocolos más prometedores que van en esta dirección está el Protocolo de iniciación de sesiones (SIP, *Session Initiation Protocol*), definido en [RFC 3261; RFC5411]. SIP es un protocolo ligero que hace lo siguiente:

- Proporciona mecanismos para establecer llamadas entre el que llama y el que es llamado a través de una red IP. Permite al que llama notificar al llamado que desea iniciar una comunicación. También permite a los participantes acordar los métodos de codificación de los datos multimedia, así como dar por terminadas las llamadas.
- Proporciona mecanismos al que llama para determinar la dirección IP actual del llamado. Los usuarios no tienen una única dirección fija porque pueden asignarse dinámicamente direcciones (mediante DHCP) y porque pueden tener múltiples dispositivos IP, teniendo cada uno de ellos una dirección IP diferente.
- Proporciona mecanismos para la gestión de llamadas tales como añadir nuevos flujos multimedia durante la llamada, cambiar el método de codificación o invitar a nuevos participantes mientras tiene lugar la llamada, además de mecanismos para la transferencia de llamadas y el mantenimiento de las mismas.

Establecimiento de una llamada con una dirección IP conocida

Para comprender la esencia de SIP, lo mejor es analizar un ejemplo concreto. En este ejemplo, Alicia se encuentra delante de su PC y desea llamar a Benito, que también está trabajando con su computadora. Los PC de Alicia y de Benito están equipados con software basado en SIP que les permite hacer y recibir llamadas telefónicas. En este ejemplo inicial, supondremos que Alicia conoce la dirección IP de la computadora de Benito. En la Figura 7.14 se ilustra el proceso SIP de establecimiento de llamadas.

En la Figura 7.14 vemos que se inicia una sesión SIP cuando Alicia envía a Benito un mensaje INVITE, que es parecido a un mensaje de solicitud HTTP. Este mensaje INVITE se envía sobre UDP al puerto bien conocido 5060 para SIP. (Los mensajes SIP también se pueden enviar sobre TCP.) El mensaje INVITE incluye un identificador para Benito

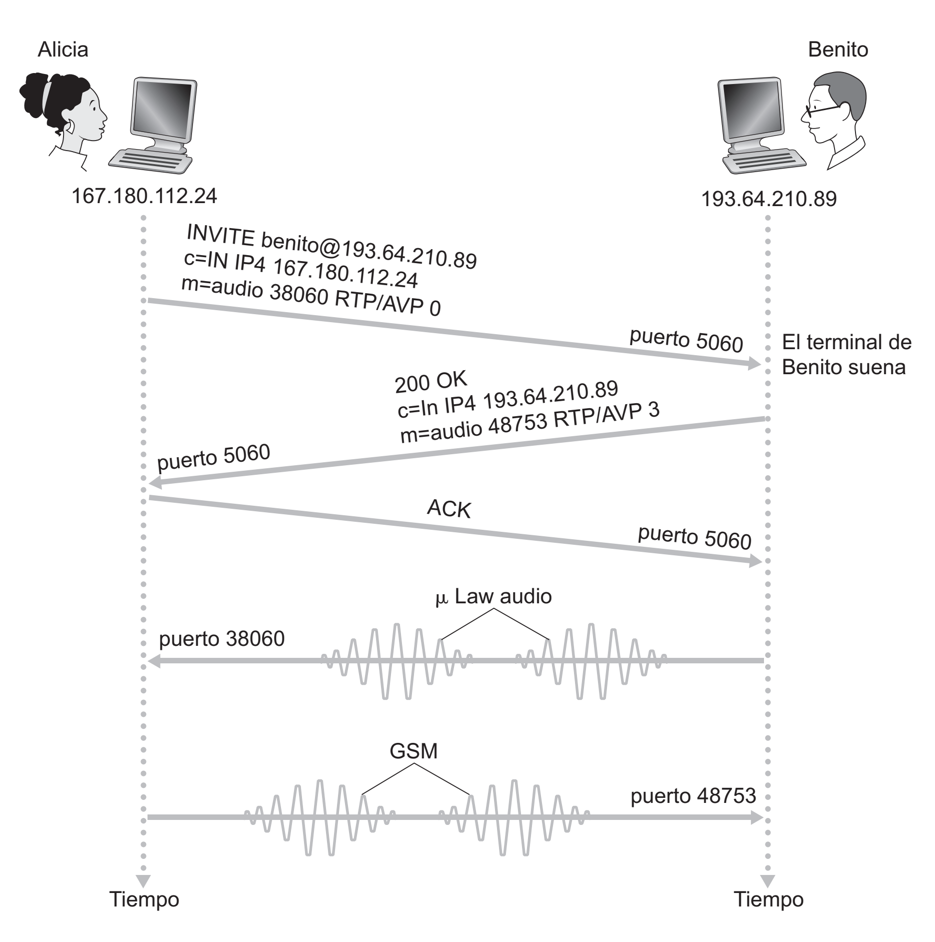


Figura 7.14 • Establecimiento de una llamada SIP cuando Alicia conoce la dirección IP de Benito.

(benito@193.64.210.89), una indicación de la dirección IP actual de Alicia, una indicación de que Alicia desea recibir audio, el cual debe codificarse en formato AVP 0 (codificación PCM μ-law) y encapsularse en RTP, y una indicación de que Alicia desea recibir los paquetes RTP en el puerto 38060. Después de recibir el INVITE de Alicia, Benito envía un mensaje de respuesta SIP, que es similar a un mensaje de respuesta HTTP. Este mensaje de respuesta SIP también se envía al puerto SIP 5060. La respuesta de Benito incluye un mensaje 200 OK y una indicación de su dirección IP, la forma de codificación y empaquetamiento que desea para recepción y su número de puerto al que deben enviarse los paquetes de audio. Observe que, en este ejemplo, Alicia y Benito van a emplear diferentes mecanismos de codificación de audio: a Alicia se le solicita que codifique su audio con GSM mientras que a Benito se le pide que codifique su audio con PCM μ-law. Una vez recibida la respuesta de Benito, Alicia envía a Benito un mensaje de reconocimiento SIP. Después de esta transacción SIP, Benito y Alicia pueden hablar. (Por comodidad visual, la Figura 7.14 muestra que Alicia habla después de Benito, pero en la realidad normalmente hablan al mismo tiempo.) Benito codificará y empaquetará el audio de la forma requerida y enviará los paquetes de audio al número de puerto 38060 en la dirección IP 167.180.112.24. Alicia

también codificará y empaquetará el audio en el formato solicitado y enviará los paquetes de audio al número de puerto 48753 en la dirección IP 193.64.210.89.

Con este sencillo ejemplo hemos aprendido una serie de características clave de SIP. En primer lugar, SIP es un protocolo fuera de banda: los mensajes SIP se envían y se reciben en sockets diferentes de los utilizados para enviar y recibir los datos multimedia. En segundo lugar, los propios mensajes SIP son mensajes legibles ASCII y son parecidos a los mensajes HTTP. Por último, SIP requiere que todos los mensajes sean reconocidos, por lo que se puede ejecutar sobre UDP o sobre TCP.

Siguiendo con este ejemplo, ahora vamos a considerar lo que ocurriría si Benito no dispone de un codec PCM μ-law para codificar el audio. En este caso, en lugar de responder con `200 OK`, Benito probablemente respondería con el mensaje `600 Not Acceptable` e incluiría en el mensaje todos los codecs que puede utilizar. Alicia elegiría entonces uno de los codecs de la lista y enviaría otro mensaje INVITE, anunciando en esta ocasión el codec elegido. Benito también podría simplemente rechazar la llamada enviando uno de los muchos posibles códigos de respuesta de rechazo. (Existen muchos códigos de este tipo, entre los que se incluyen "ocupado", "ausente", "pago requerido" y "prohibido".)

Direcciones SIP

En el ejemplo anterior, la dirección SIP de Benito es `sip:benito@193.64.210.89`. Sin embargo, es deseable que muchas (si no la mayoría) direcciones SIP se parezcan a direcciones de correo electrónico. Por ejemplo, la dirección de Benito podría ser `sip:benito@dominio.com`. Cuando el dispositivo SIP de Alicia envía un mensaje INVITE, el mensaje incluiría esta dirección similar a una dirección de correo electrónico; la infraestructura SIP enrutaría entonces el mensaje al dispositivo IP que Benito esté utilizando en ese momento (como veremos más adelante). Otro posible formato de la dirección SIP podría ser el número de teléfono de Benito o simplemente su nombre y apellidos (suponiendo siempre que fueran únicos).

Una característica interesante de las direcciones SIP es que pueden incluirse en páginas web, al igual que se incluyen las direcciones de correo electrónico en las páginas web con el URL mailto. Por ejemplo, suponga que Benito tiene una página web personal y que desea proporcionar un medio a sus visitantes para que le llamen. Benito puede simplemente incluir el URL `sip:benito@dominio.com` y, de este modo, cuando un visitante haga clic en el URL, la aplicación SIP del dispositivo del visitante se ejecuta y envía un mensaje INVITE a Benito.

Mensajes SIP

En esta breve introducción al protocolo SIP no vamos a tratar todas las cabeceras y tipos de mensajes SIP. En lugar de ello vamos a examinar brevemente el mensaje SIP INVITE junto con unas pocas líneas de cabecera comunes. Supongamos de nuevo que Alicia desea iniciar una llamada de teléfono IP con Benito y en esta ocasión Alicia conoce la dirección SIP de Benito, `benito@dominio.com`, pero no conoce la dirección IP del dispositivo que Benito está utilizando actualmente. El mensaje de Alicia sería similar al siguiente:

```
INVITE sip:benito@dominio.com SIP/2.0
Via: SIP/2.0/UDP 167.180.112.24
From: sip:alicia@hereway.com
To: sip:benito@dominio.com
```

```
Call-ID: a2e3a@pigeon.hereway.com
Content-Type: application/sdp
Content-Length: 885

c=IN IP4 167.180.112.24
m=audio 38060 RTP/AVP 0
```

La línea INVITE incluye la versión de SIP, al igual que en un mensaje de solicitud HTTP. Cuando un mensaje SIP atraviesa un dispositivo SIP (incluyendo el dispositivo que origina el mensaje) asocia una línea de cabecera `Via`, que indica la dirección IP del dispositivo. (Veremos enseguida que el mensaje INVITE típico atraviesa muchos dispositivos SIP antes de llegar a la aplicación SIP del llamado.) De formar similar a un mensaje de correo electrónico, el mensaje SIP incluye una línea de cabecera `From` y una línea de cabecera `To`. El mensaje incluye también la línea de cabecera `Call-ID`, que identifica de forma unívoca la llamada (de forma similar al `message-ID` de un correo electrónico). También incluye la línea de cabecera `Content-Type`, que define el formato utilizado para describir el contenido del mensaje SIP, e incluye la línea de cabecera `Content-Length`, que proporciona la longitud en bytes del contenido del mensaje. Por último, después de un retorno de carro y un salto de línea, se incluye el contenido del mensaje. En este caso, el contenido proporciona información acerca de la dirección IP de Alicia y de cómo desea ésta recibir el audio.

Traducción de nombres y localización de usuarios

En el ejemplo de la Figura 7.14 hemos supuesto que el dispositivo SIP de Alicia conocía la dirección IP en la que podría contactar a Benito. Pero esta suposición es bastante poco realista, no sólo porque las direcciones IP a menudo son asignadas dinámicamente mediante DHCP, sino también porque Benito dispone de varios dispositivos IP (por ejemplo, diversos dispositivos en su domicilio, en el trabajo y en el coche). Por tanto, ahora vamos a suponer que Alicia sólo conoce la dirección de correo electrónico de Benito, `benito@dominio.com`, y que esta misma dirección se utiliza para las llamadas basadas en SIP. En este caso, Alicia tiene que obtener la dirección IP del dispositivo que está utilizando actualmente el usuario `benito@dominio.com`. Para determinar esto, Alicia crea un mensaje INVITE que comienza con `INVITE benito@dominio.com SIP/2.0` y envía este mensaje a un **proxy SIP**. El proxy contestará con una respuesta SIP que podría incluir la dirección IP del dispositivo que actualmente está utilizando `benito@dominio.com`. De forma alternativa, la respuesta podría incluir la dirección IP del buzón de voz de Benito, o podría incluir un URL de una página web (que diga "¡Benito está durmiendo. No molestar!"). Además, el resultado devuelto por el proxy podría depender de quién hace la llamada: si la llamada es de la esposa de Benito, aceptará la llamada y suministrará su dirección IP; si la llamada la hace la suegra de Benito, podría responder con el URL que apunta a la página web de ¡Estoy durmiendo!

Es posible que en este momento se esté preguntando cómo puede el servidor proxy determinar la dirección IP actual para `benito@dominio.com`. Para responder a esta pregunta, primero tenemos que decir algunas cosas acerca de otro dispositivo SIP: el **registrador SIP**. Todos los usuarios SIP tienen un asociado registrador. Cuando un usuario ejecuta una aplicación SIP en un dispositivo, la aplicación envía un mensaje de registro SIP al registrador, informándole de su dirección IP actual. Por ejemplo, cuando Benito ejecuta su aplicación SIP en su PDA, la aplicación enviará un mensaje con las siguientes líneas:

```
REGISTER sip:dominio.com SIP/2.0
Via: SIP/2.0/UDP 193.64.210.89
From: sip:benito@dominio.com
To: sip:benito@dominio.com
Expires: 3600
```

El registrador de Benito mantiene su dirección IP actual. Cuando Benito cambia a un dispositivo SIP nuevo, éste envía un nuevo mensaje de registro indicando la nueva dirección IP. Además, si Benito permanece en el mismo dispositivo durante un periodo de tiempo largo, el dispositivo enviará mensajes de registro de refresco, indicando que la dirección IP enviada más recientemente continúa siendo válida. (En el ejemplo anterior, los mensajes de refresco necesariamente eran enviados cada 3.600 segundos con el fin de mantener la dirección en el servidor registrador.) Merece la pena comentar que el registrador es análogo a un servidor de nombres DNS autoritativo: el servidor DNS traduce los nombres fijos de host a direcciones IP fijas y el registrador SIP traduce los identificadores fijos de personas (por ejemplo, `benito@dominio.com`) a direcciones IP dinámicas. A menudo los registradores SIP y los proxies SIP se ejecutan en el mismo host.

Examinemos ahora cómo el servidor proxy de Alicia obtiene la dirección IP actual de Benito. En la exposición anterior hemos visto que el servidor proxy simplemente necesita reenviar el mensaje INVITE de Alicia al registrador/proxy de Benito. El registrador/proxy podría entonces reenviar el mensaje al dispositivo SIP actual de Benito. Por último, una vez que Benito ha recibido el mensaje INVITE de Alicia, podría enviar una respuesta SIP a Alicia.

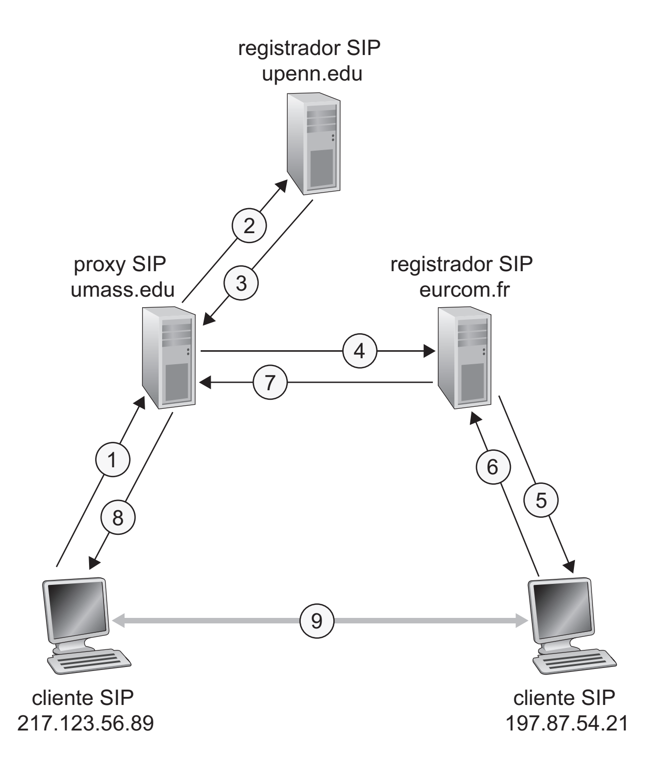


Figura 7.15 • Iniciación de sesión que implica proxies y registradores SIP.

Por ejemplo, considere la Figura 7.15, en la que juan@umass.edu, que actualmente está trabajando en 217.123.56.89, desea iniciar una sesión de Voz sobre IP (VoIP) con katy@upenn.edu, que se encuentra en este momento trabajando en 197.87.54.21. Los pasos que se llevan a cabo son los siguientes: (1) Juan envía un mensaje INVITE al proxy SIP umass. (2) El proxy realiza una búsqueda DNS en el registrador SIP upenn.edu (no mostrado en el diagrama) y luego reenvía el mensaje al servidor registrador. (3) Puesto que katy@upenn.edu todavía no está registrada en el registrador upenn, éste envía una respuesta de redirección que indica que debería probar katy@eurecom.fr. (4) El proxy umass envía un mensaje INVITE al registrador SIP eurecom. (5) El registrador eurecom conoce la dirección IP de katy@eurecom.fr y reenvía el mensaje INVITE al host 197.87.54.21, que está ejecutando el cliente SIP de Katy. (6-8) Se devuelve al cliente SIP 217.123.56.89 una respuesta SIP a través de los registradores/proxies. (9) Los datos multimedia se envían directamente entre los dos clientes. (También existe un mensaje de reconocimiento SIP, que no se muestra en la figura.)

Nuestra exposición acerca de SIP se ha centrado en el proceso de iniciación de llamada para el caso de llamadas de voz. SIP, aunque en general es un protocolo de señalización para el inicio y la terminación de llamadas, puede utilizarse para videoconferencia, así como para sesiones basadas en texto. De hecho, SIP se ha convertido en un componente fundamental en muchas aplicaciones de mensajería instantánea. Animamos a los lectores que deseen aprender más acerca de SIP a visitar el sitio web de Henning Schulzrinne [Schulzrinne-SIP 2009]. En particular, en este sitio encontrará software de código abierto para clientes y servidores SIP [SIP Software 2009].

7.4.4 H.323

Como alternativa a SIP, H.323 es un estándar popular para audio y videoconferencia en tiempo real entre sistemas terminales de Internet. Como se muestra en la Figura 7.16, el estándar también cubre cómo los sistemas terminales conectados a Internet se comunican con teléfonos conectados a redes telefónicas de conmutación de circuitos. (SIP también hace esto, aunque no lo vamos a ver aquí.) El canalizador H.323 es un dispositivo similar a un registrador SIP.

El estándar H.323 es una especificación paraguas que incluye las siguientes especificaciones:

- Una especificación acerca de cómo los puntos terminales negocian los sistemas de codificación comunes de audio/vídeo. Puesto que H.323 soporta una serie de estándares de codificación de audio y vídeo, se necesita un protocolo que permita la comunicación entre los puntos terminales con el fin de acordar un sistema de codificación común.
- Una especificación acerca de cómo se encapsulan y se envían los fragmentos de audio y vídeo a través de la red. En particular, H.323 obliga el uso de RTP para este propósito.
- Una especificación acerca de cómo los puntos terminales se comunican con sus respectivos canalizadores.
- Una especificación acerca de cómo los teléfonos Internet se comunican a través de una pasarela con teléfonos ordinarios de la red pública de telefonía conmutada.

Como mínimo, cada punto terminal H.323 *debe* soportar el estándar de compresión de voz G.711. G.711 utiliza PCM para generar voz digitalizada a 56 o 64 kbps. Aunque H.323

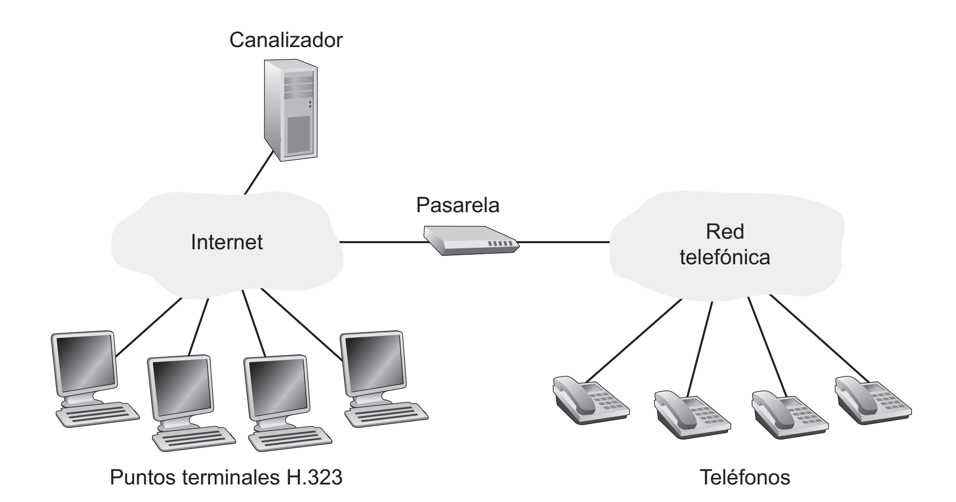


Figura 7.16 • Sistemas terminales H.323 conectados a Internet pueden comunicarse con teléfonos conectados a una red telefónica de circuitos conmutados.

requiere que todos los puntos terminales permitan voz (a través de G.711), las capacidades de vídeo son opcionales. Dado que el soporte de vídeo es opcional, los fabricantes de terminales pueden vender terminales de voz más simples, así como terminales más complejos que soporten tanto audio como vídeo. Las capacidades de vídeo para un punto terminal H.323 son opcionales. Sin embargo, si un punto terminal soporta vídeo, entonces tiene que soportar (como mínimo) el estándar de vídeo QCIF H.261 (176 $\times$144 píxeles).

H.323 es un estándar muy amplio que, además de los estándares y protocolos descritos anteriormente, impone el uso del protocolo de control H.245, de un canal de señalización Q.931 y de un protocolo RAS para el registro con el canalizador.

Concluimos esta sección examinando algunas de las diferencias más importantes entre H.323 y SIP.

- H.323 es una serie completa y verticalmente integrada de protocolos para conferencias multimedia: señalización, registro, control de admisión, transporte y codecs (codificadores-decodificadores).
- Por el contrario, SIP únicamente controla el inicio y gestión de las sesiones, constando de un único componente. SIP funciona con RTP pero no es obligatorio. Funciona con los codecs de voz G.711 y los codecs de vídeo QCIF H.261, pero tampoco son obligatorios. Se puede combinar con otros protocolos y servicios.
- H.323 procede de la ITU (telefonía), mientras que SIP procede de IETF y toma muchos conceptos de la Web, DNS y el correo electrónico de Internet.
- H.323, al ser un estándar paraguas, es grande y complejo. SIP utiliza el principio de sencillez KISS (*keep it simple, stupid*).

Si desea ver una excelente exposición acerca de H.323, SIP y VoIP en general, consulte [Hersent 2000].

7.5 Múltiples clases de servicios

En las secciones anteriores hemos visto cómo las aplicaciones multimedia pueden utilizar los números de secuencia, las marcas de tiempo, FEC, RTP y H.323 en la red Internet actual. Las redes CDN representan una solución global de sistema para distribuir contenido multimedia. Pero, ¿son estas técnicas por sí mismas suficientes para soportar aplicaciones fiables y robustas tales como un servicio de telefonía IP que sea equivalente a la red telefónica actual? Antes de responder a esta pregunta, recordemos una vez más que la red Internet actual proporciona un servicio de entrega de mejor esfuerzo a todas sus aplicaciones; es decir, no hace ninguna promesa relativa a la calidad del servicio que recibirá una aplicación. La aplicación recibirá aquel nivel de rendimiento (por ejemplo, pérdida y retardos de paquetes terminal a terminal) que la red sea capaz de proporcionar en cada momento. Recuerde también que la red Internet pública actual no permite que las aplicaciones multimedia sensibles al retardo soliciten ningún tratamiento especial. Dado que todos los paquetes, incluyendo los paquetes de audio y de vídeo sensibles al retardo, son tratados de la misma forma en los routers, la calidad de una llamada de teléfono IP activa puede arruinarse si existe el suficiente tráfico que interfiera (es decir, congestión de red) como para aumentar de forma notable el retardo y las pérdidas que afectan a la llamada.

Pero si el objetivo es el de proporcionar un modelo de servicio que ofrezca algo más que el servicio de mejor esfuerzo de la red Internet actual (que vale para todo), ¿qué tipo de servicio hay que proporcionar exactamente? Un modelo de servicio simple mejorado consistiría en clasificar el tráfico en clases y proporcionar diferentes niveles de servicio a esas distintas clases de tráfico. Por ejemplo, un ISP puede desear proporcionar una clase de servicio mejor al tráfico de teleconferencia o de voz sobre IP sensibles al retardo (¡y cobrar más por este servicio!) que al tráfico elástico como el de FTP o HTTP. Todos estamos familiarizados con las distintas clases de servicio en nuestra vida cotidiana: los pasajeros de primera clase de una línea aérea obtienen un mejor servicio que los pasajeros de la clase business, quienes a su vez obtienen un mejor servicio que el que se proporciona a los pasajeros de la clase turista; las personas VIP pueden entrar de forma inmediata a los eventos mientras que los demás tienen que esperar en fila; en algunos países las personas mayores son tremendamente respetadas y se les ofrece los asientos de honor y lo más exquisito de los alimentos en la mesa.

Es importante observar que tal servicio diferenciado se proporciona entre *agregados* de tráfico; es decir, entre clases de tráfico, no entre conexiones individuales. Por ejemplo, todos los pasajeros de primera clase son tratados igual (ningún pasajero de primera clase recibe un tratamiento mejor que cualquier otro pasajero también de primera clase), al igual que todos los paquetes VoIP recibirán el mismo tratamiento dentro de la red, independientemente de la conexión terminal a terminal concreta a la que pertenezcan. Como veremos, al tratar con un número pequeño de agregados de tráfico en lugar de con una gran cantidad de conexiones individuales, los nuevos mecanismos de red requeridos para proporcionar un servicio de entrega más efectivo que el de mejor esfuerzo pueden ser relativamente simples.

Evidentemente, los primeros diseñadores de Internet tenían esta idea de múltiples clases de servicios en mente. Recuerde el campo Tipo de servicio (ToS) de la cabecera IPv4 en la Figura 4.13. IEN123 [ISI 1979] describe el campo ToS también presente en un antecesor del datagrama IPv4 de la siguiente manera: “El campo Tipo de servicio proporciona una indicación de los parámetros abstractos de la calidad del servicio deseado. Estos parámetros se emplean para guiar la selección de los parámetros del servicio al transmitir un datagrama

a través de una red concreta. Varias redes ofrecen mecanismos de precedencia de servicio, que tratan en cierto modo el tráfico con precedencia alta como si fuera más importante que el resto del tráfico.” Incluso hace tres décadas, ¡la visión de proporcionar diferentes niveles de servicio a los diferentes niveles de tráfico estaba clara! Sin embargo, nos ha llevado todo ese tiempo el conseguir llevar a la práctica dicha visión.

Comenzaremos nuestro estudio en la Sección 7.5.1 considerando varios escenarios que motivarán la necesidad de mecanismos específicos que den soporte a múltiples clases de servicio. A continuación nos ocuparemos de dos temas importantes: la planificación en el nivel de enlace y la vigilancia/clasificación de paquetes en la Sección 7.5.2. En la Sección 7.5.3 cubriremos los servicios diferenciados (Diffserv): el estándar actual de Internet para proporcionar servicios diferenciados.

7.5.1 Escenarios

La Figura 7.17 muestra un escenario de red simple. Suponga que fluyen dos paquetes de aplicación originados en los hosts H1 y H2 de una LAN y que están destinados a los hosts H3 y H4 de otra LAN. Los routers de las dos redes LAN están conectados mediante un enlace a 1,5 Mbps. Suponemos que las velocidades de las redes LAN son significativamente más altas que 1,5 Mbps y vamos a fijarnos en la cola de salida del router R1; es aquí donde se producirá el retardo y la pérdida de paquetes si la velocidad agregada de transmisión de H1 y H2 excede los 1,5 Mbps. Consideremos ahora varios escenarios, que nos van a presentar la necesidad de mecanismos específicos para dar soporte a múltiples clases de servicios.

Escenario 1: una aplicación de audio de 1 Mbps y una transferencia FTP

El escenario 1 se ilustra en la Figura 7.18. En este caso, una aplicación de audio de 1 Mbps (por ejemplo, una llamada de audio con calidad de CD) comparte el enlace a 1,5 Mbps entre R1 y R2 con una aplicación FTP que está transfiriendo un archivo desde H2 a H4. Con el

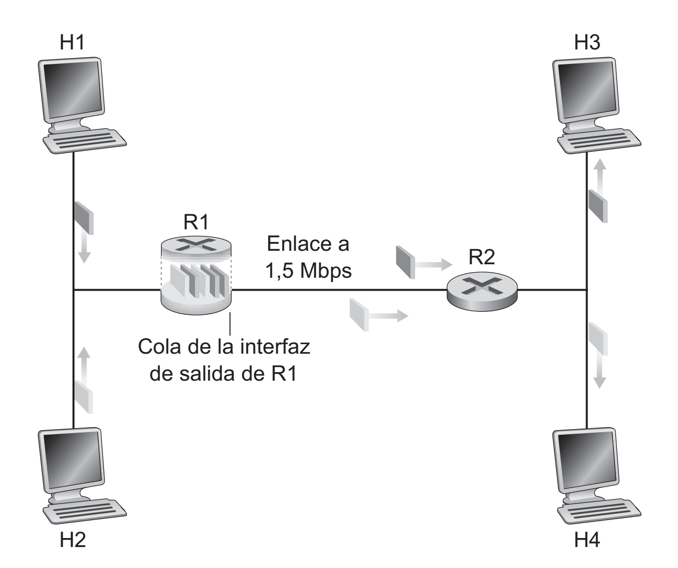


Figura 7.17 • Una red simple con dos aplicaciones.

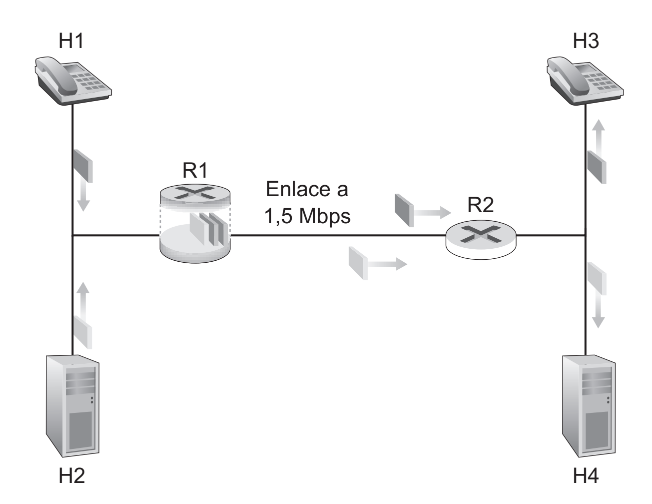


Figura 7.18 • Competencia entre aplicaciones de audio y FTP.

servicio de entrega del mejor esfuerzo de Internet, los paquetes de audio y FTP se mezclan en la cola de salida de R1 y (normalmente) se transmiten siguiendo el orden primero en entrar primero en salir (FIFO, *First-In-First-Out*). En este escenario, una ráfaga de paquetes procedentes del origen FTP podría potencialmente llenar la cola, haciendo que los paquetes de audio IP se retardaran excesivamente o se perdieran a causa de un desbordamiento del buffer de R1. ¿Cómo podemos resolver este potencial problema? Dado que la aplicación FTP no tiene restricciones de tiempo, nuestra intuición puede llevarnos a pensar que debería proporcionarse una prioridad estricta a los paquetes de audio en R1. Aplicando esta disciplina de planificación con prioridad estricta, un paquete de audio que se encuentra en el buffer de salida de R1 siempre debería transmitirse antes que cualquier paquete FTP que se encuentre en este mismo buffer. El enlace entre R1 y R2 sería entonces como un enlace dedicado a 1,5 Mbps para el tráfico de audio y el tráfico FTP emplearía dicho enlace sólo cuando no hubiera tráfico de audio en la cola.

Con el fin de que R1 distinga los paquetes de audio de los paquetes FTP que tiene en su cola, cada uno de los paquetes debe marcarse como perteneciente a una de esas dos clases de tráfico. Éste era el objetivo original del campo Tipo de servicio (ToS) de IPv4. Aunque pueda parecer obvio, éste es nuestro primer principio básico en el que se fundamentan los mecanismos necesarios para proporcionar múltiples clases de tráfico:

> **Principio 1:** el marcado de los paquetes permite a un router diferenciar entre paquetes pertenecientes a distintas clases de tráfico.

Escenario 2: una aplicación de audio de 1 Mbps y una transferencia FTP con prioridad alta

Nuestro segundo escenario es ligeramente diferente del escenario 1. Suponga ahora que el usuario FTP ha comprado a su ISP un servicio “platinum” (es decir, de alto precio), mientras que el usuario de audio ha adquirido un servicio Internet de bajo presupuesto y barato, cuyo coste es una minúscula fracción del coste del servicio platinum. En este caso, ¿debería darse prioridad a los paquetes de audio del usuario que ha adquirido un servicio

barato frente a los paquetes FTP? Posiblemente no. En este caso, parece más razonable diferenciar los paquetes basándose en la dirección IP del emisor. En general, vemos que es necesario que los routers *clasifiquen* los paquetes de acuerdo con determinados criterios. Teniendo esto en cuenta, tenemos que modificar ligeramente el principio 1:

Principio 1 (modificado): la clasificación de paquetes permite a los routers diferenciar entre paquetes que pertenecen a distintas clases de tráfico.

El marcado explícito de los paquetes es una forma que permite diferenciarlos. Sin embargo, la marca que transporta el paquete, por sí misma, no implica que el paquete recibirá una determinada calidad de servicio. El marcado es simplemente un *mecanismo* para distinguir los paquetes. La manera en que un router distingue entre paquetes tratándolos de forma diferente es una decisión de *vigilancia.*

Escenario 3: una aplicación de audio con mal comportamiento y una transferencia FTP

Suponga ahora que, de alguna manera (usando mecanismos que estudiaremos en las siguientes secciones), el router sabe que debe dar prioridad a los paquetes procedentes de la aplicación de audio a 1 Mbps. Puesto que la velocidad del enlace de salida es de 1,5 Mbps, incluso aunque los paquetes FTP tengan una prioridad menor todavía recibirán, como promedio, un servicio de transmisión de 0,5 Mbps. Pero, ¿qué ocurre si la aplicación de audio comienza a enviar paquetes a una velocidad de 1,5 Mbps o superior (bien maliciosamente o debido a un error de la aplicación)? En este caso, los paquetes FTP no recibirán ningún servicio del enlace R1 a R2. Podrían producirse problemas similares si varias aplicaciones (por ejemplo, varias llamadas de audio), todas ellas con la misma prioridad, tuvieran que compartir el ancho de banda de un enlace; un flujo no conforme podría degradar y arruinar el rendimiento de los restantes flujos. Idealmente, es deseable un grado de *aislamiento* entre clases de tráfico y también posiblemente entre flujos de una misma clase de tráfico, con el fin de proteger a cada uno de los flujos de aquellos flujos que presentan un comportamiento erróneo. El concepto de proteger los flujos individuales de una determinada clase frente a las otras contradice nuestra observación anterior acerca de que los paquetes de todos los flujos pertenecientes a una clase deberían ser tratados del mismo modo. En la práctica, los paquetes de una clase son de hecho tratados del mismo modo en los routers del núcleo de la red. Sin embargo, en la frontera de la red los paquetes de un determinado flujo pueden ser monitorizados para garantizar que la velocidad agregada de un flujo individual no excede un cierto valor.

Estas consideraciones nos llevan a nuestro segundo principio:

Principio 2: es deseable proporcionar un grado de aislamiento entre las clases de tráfico y entre los flujos, de manera que una clase o un flujo no se vea afectado de forma adversa por otro que tiene un comportamiento erróneo.

En la siguiente sección vamos a examinar varios mecanismos específicos para proporcionar este aislamiento entres clases de tráfico o flujos. Debemos comentar aquí que es posible adoptar dos enfoques. En primer lugar, como se muestra en la Figura 7.19, es posible vigilar el tráfico. Si una clase de tráfico o flujo tiene que satisfacer ciertos criterios (por ejemplo, que el flujo de audio no exceda la velocidad de pico de 1 Mbps), entonces puede

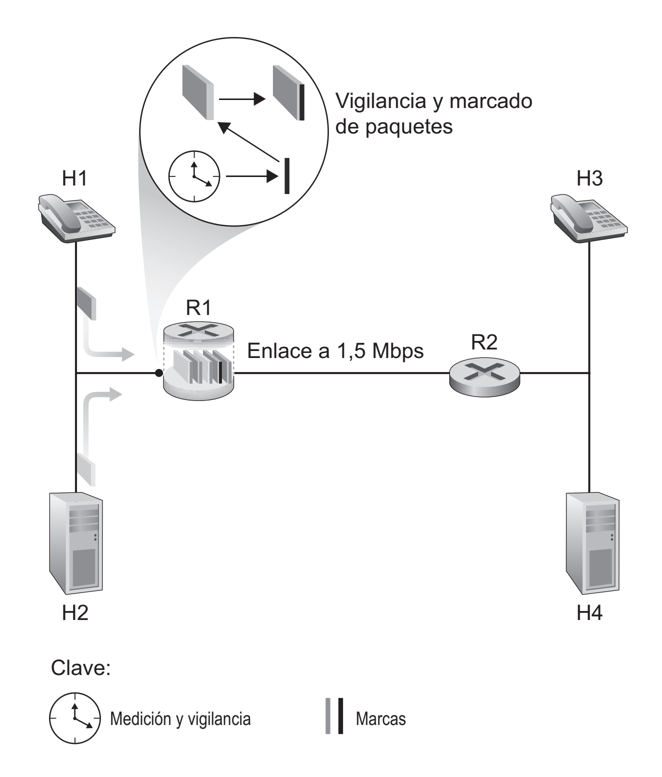


Figura 7.19 • Vigilancia (y marcado) de los flujos de tráfico de audio y FTP.

utilizarse un mecanismo de vigilancia con el fin de garantizar que esos criterios son observados. Si la aplicación que se está monitorizando presenta un mal comportamiento, entonces el mecanismo de vigilancia llevará a cabo una cierta acción (por ejemplo, descartar o retardar los paquetes que están violando los criterios), de modo que el tráfico que realmente entre en la red cumpla los criterios. El mecanismo de goteo (una cubeta con pérdidas) que examinaremos en la siguiente sección es quizá el mecanismo de vigilancia más ampliamente utilizado. En la Figura 7.19, el mecanismo de clasificación y marcado de paquetes (Principio 1) y el mecanismo de vigilancia (Principio 2) están localizados en la frontera de la red, bien en el sistema terminal o bien en un router de frontera.

Un enfoque alternativo para proporcionar aislamiento entre clases de tráfico o flujos es que el mecanismo de planificación de paquetes a nivel de enlace asigne explícitamente una cantidad fija de ancho de banda del enlace a cada clase o flujo. Por ejemplo, al flujo de audio podría asignársele 1 Mbps en R1 y al flujo FTP se le podría asignar 0,5 Mbps. En este caso, los flujos de audio y FTP ven un enlace lógico con una capacidad de 1,0 y 0,5 Mbps, respectivamente, como se muestra en la Figura 7.20.

Con un cumplimiento estricto de la asignación de ancho de banda a nivel de enlace, una clase o un flujo sólo puede usar la cantidad de ancho de banda que ha sido asignada; en concreto, no puede utilizar ancho de banda que no esté siendo actualmente empleado por otros. Por ejemplo, si el flujo de audio se silencia (por ejemplo, si el que habla hace una pausa y no genera paquetes de audio), el flujo FTP seguirá sin poder transmitir a más de 0,5 Mbps a través del enlace de R1 a R2, incluso aunque la asignación de ancho de banda de 1 Mbps del flujo de audio no esté siendo utilizada en ese momento. Por tanto, es de-

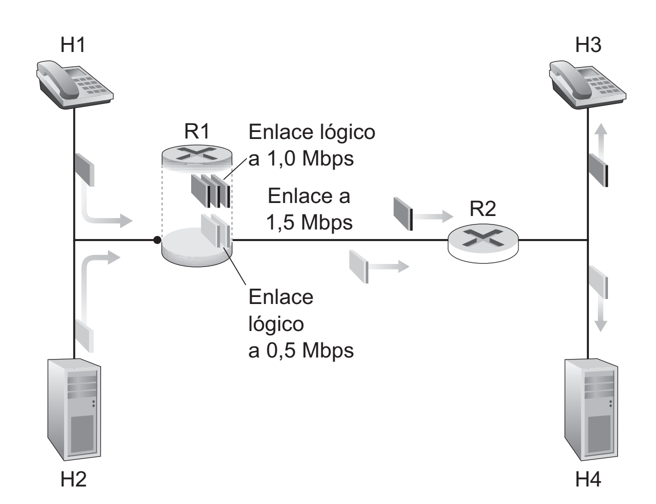


Figura 7.20 • Aislamiento lógico de los flujos de las aplicaciones de audio y FTP.

seable utilizar el ancho de banda de la forma más eficiente posible, permitiendo a una clase de tráfico o flujo utilizar el ancho de banda no utilizado por otros en cualquier instante dado. Estas consideraciones nos llevan a nuestro tercer principio:

> **Principio 3:** mientras se proporciona aislamiento entre clases o flujos, es deseable utilizar los recursos (por ejemplo, el ancho de banda del enlace y los buffers) de la forma más eficiente posible.

7.5.2 Mecanismos de planificación y vigilancia

Ahora que hemos identificado los principios en que se fundamentan los mecanismos necesarios para proporcionar las distintas clases de servicios, vamos a considerar en detalle dos de los mecanismos más importantes: la planificación y la vigilancia.

Mecanismos de planificación

Recuerde que en las Secciones 1.3 y 4.3 hemos visto que los paquetes que pertenecen a varios flujos de red se multiplexan y se ponen en cola para su transmisión en los buffers de salida asociados con un enlace. La forma en que los paquetes puestos en cola son seleccionados para su transmisión a través del enlace se conoce como **disciplina de planificación de enlace**. Vamos a considerar ahora más detalladamente varias de las disciplinas de planificación de enlace más importantes.

Primero en entrar-primero en salir (FIFO)

La Figura 7.21 muestra las abstracciones del modelo de cola para la disciplina de planificación de enlace FIFO. Los paquetes que llegan a la cola de salida del enlace esperan para ser transmitidos si el enlace actualmente está ocupado transmitiendo otro paquete. Si no hay suficiente espacio en el buffer como para almacenar el paquete que ha llegado, la **política de eliminación de paquetes** de la cola determina si el paquete será eliminado (paquete perdido) o si se serán eliminados otros paquetes de la cola para hacer espacio al paquete recién

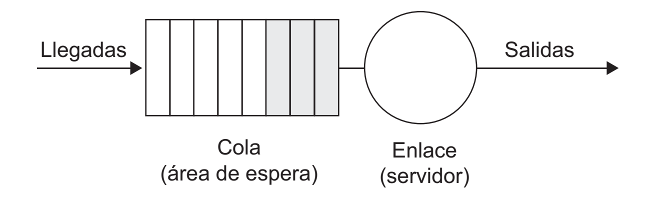


Figura 7.21 • Abstracción de la cola FIFO.

llegado. En la exposición que sigue vamos a ignorar la eliminación de paquetes. Cuando un paquete se transmite completamente a través del enlace de salida (es decir, recibe servicio) se elimina de la cola.

La disciplina de planificación FIFO, también conocida como FCFS (*First-Come-First-Served*, primero en entrar, primero en ser servido), selecciona los paquetes para su transmisión a través del enlace en el mismo orden en que han llegado a la cola del enlace de salida. Todos estamos familiarizados con las colas FIFO gracias a las paradas de los autobuses (especialmente en Inglaterra, donde las colas parecen haber sido perfeccionadas) o a otros centros de servicios, donde los clientes que llegan se colocan al final de la fila, permaneciendo en orden y luego van siendo servidos a medida que llegan al principio de la fila.

La Figura 7.22 muestra el funcionamiento de la cola FIFO. La llegada de los paquetes está indicada mediante flechas numeradas por encima de la línea de tiempo superior, indicando el número de orden en el que ha llegado el paquete. La salida de cada uno de los paquetes se muestra por debajo de la línea de tiempo inferior. El instante en el que un paquete recibe servicio (está siendo transmitido) se indica mediante el rectángulo sombreado situado entre las dos líneas de tiempo. En la disciplina FIFO los paquetes salen en el mismo orden en el que llegaron. Observe que después de la salida del paquete 4 el enlace permanece inactivo (ya que los paquetes 1 hasta 4 han sido transmitidos y eliminados de la cola) hasta la llegada del paquete 5.

Colas con prioridad

En las **colas con prioridad** los paquetes que llegan al enlace de salida se clasifican en clases de prioridad en la cola de salida, como se muestra en la Figura 7.23. Como se ha visto en la sección anterior, la clase de prioridad de un paquete puede depender de una marca explícita

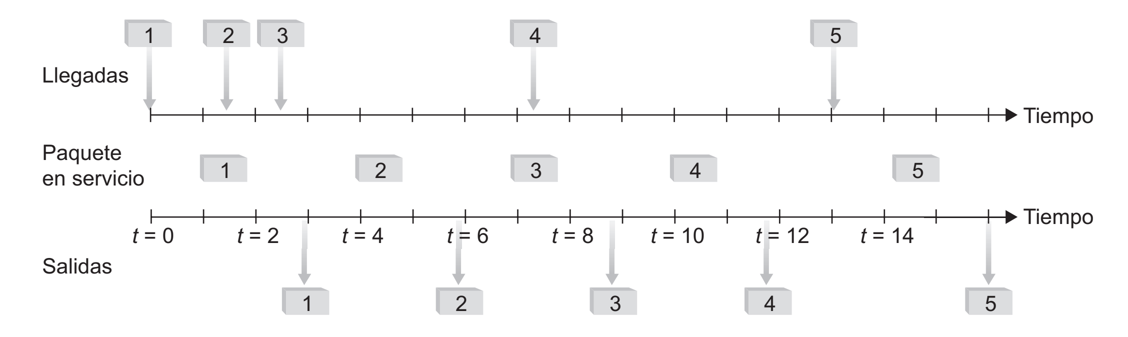


Figura 7.22 • La cola FIFO en funcionamiento.

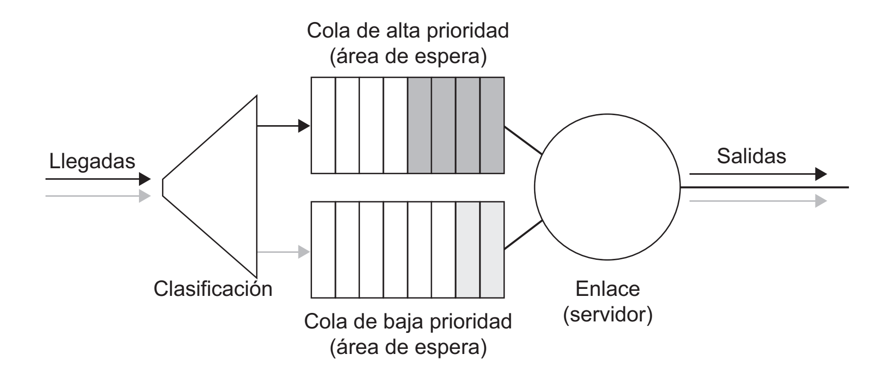


Figura 7.23 • Modelo de colas con prioridad.

que transporta en su cabecera (por ejemplo, el valor de los bits ToS de un paquete IPv4), de su dirección IP de origen o de destino, su número de puerto de destino u otro criterio. Normalmente, cada clase de prioridad tiene su propia cola. Al seleccionar un paquete para su transmisión, la disciplina de colas con prioridad transmitirá un paquete de la clase con la prioridad más alta que tenga una cola no vacía (es decir, que tenga paquetes para ser transmitidos). La elección entre paquetes *con la misma clase de prioridad* normalmente se realiza aplicando la disciplina FIFO.

La Figura 7.24 ilustra el funcionamiento de una cola con dos clases de prioridad. Los paquetes 1, 3 y 4 pertenecen a la clase con prioridad alta y los paquetes 2 y 5 pertenecen a la clase con prioridad baja. El paquete 1 llega y se encuentra con que el enlace está inactivo, por lo que inicia la transmisión. Durante la transmisión del paquete 1 llegan los paquetes 2 y 3 y se colocan, respectivamente, en las colas de baja y alta prioridad. Después de la transmisión del paquete 1 se selecciona el paquete 3 (un paquete con prioridad alta) para su transmisión antes que el paquete 2 (incluso aunque haya llegado antes, ya que es un paquete con baja prioridad). Al terminar la transmisión del paquete 3 se inicia la transmisión del paquete 2. El paquete 4 (un paquete con prioridad alta) llega durante la transmisión del paquete 2 (un paquete con prioridad baja). En una disciplina de colas con prioridad no expropiativa, la transmisión de un paquete no se interrumpe una vez que ésta ha comenzado. En este caso, el paquete 4 se pone en cola para ser transmitido y comienza a ser enviado una vez que la transmisión del paquete 2 ha sido completada.

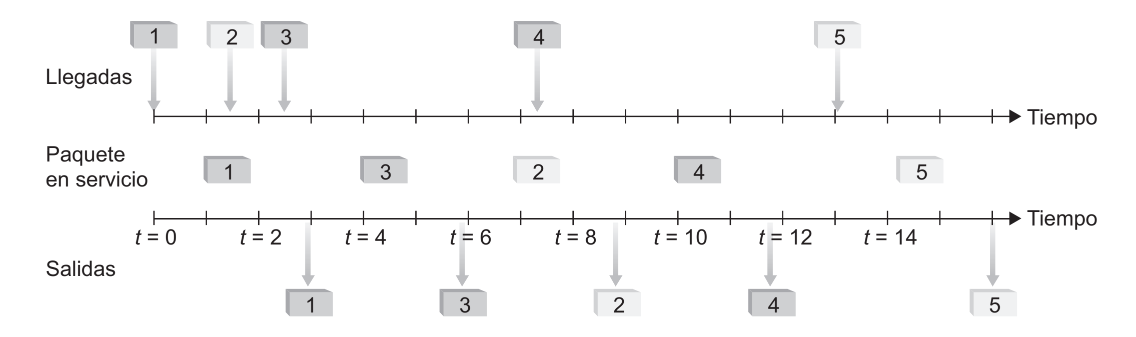


Figura 7.24 • Funcionamiento de la cola con prioridad.

Colas de turno rotatorio y colas equitativas ponderadas (WFQ)

En la **disciplina de colas de turno rotatorio (*round robin*)** los paquetes se clasifican en clases al igual que en la colas con prioridad. Sin embargo, en lugar de existir un servicio de prioridad estricta entre clases, un planificador de turno rotatorio alterna el servicio entre clases. En la forma más simple de la planificación por turno rotatorio se transmite un paquete de clase 1 seguido de un paquete de clase 2, seguido de un paquete de clase 1, seguido de un paquete de clase 2, y así sucesivamente. Una disciplina de colas conservadora nunca permitirá que el enlace permanezca inactivo siempre que haya paquetes (de cualquier clase) esperando a ser transmitidos. Una **disciplina de colas de turno rotatorio conservadora** busca un paquete de una determinada clase, pero si no lo encuentra, comprueba inmediatamente la siguiente clase de la secuencia de turno rotatorio.

La Figura 7.25 ilustra el funcionamiento de una cola por turno rotatorio de dos clases. En este ejemplo, los paquetes 1, 2 y 4 pertenecen a la clase 1 y los paquetes 3 y 5 pertenecen a la clase 2. El paquete 1 comienza a ser transmitido de forma inmediata nada más llegar a la cola de salida. Los paquetes 2 y 3 llegan mientras se está transmitiendo el paquete 1 y por tanto tienen que ponerse en cola hasta poder ser transmitidos. Una vez que el paquete 1 se ha transmitido, el planificador del enlace busca un paquete de clase 2, por lo que transmite el paquete 3. Después de transmitir este paquete, el planificador busca un paquete de clase 1 y transmite el paquete 2. Una vez transmitido el paquete 2 sólo queda en la cola el paquete 4, el cual es transmitido inmediatamente después del paquete 2.

Una abstracción generalizada de las colas de turno rotatorio que ha encontrado un considerable uso en las arquitecturas QoS es la denominada **disciplina de cola equitativa ponderada (WFQ, *Weighted Fair Queuing*)** [Demers 1990; Parekh 1993]. En la Figura 7.26 se ilustran las colas WFQ. Los paquetes que llegan se clasifican y se ponen en cola en el área de espera de la clase apropiada. Como en la planificación por turno rotatorio, un planificador WFQ dará servicio a las clases siguiendo un orden circular: primero dará servicio a la clase 1, luego a la clase 2, después a la clase 3 y luego (suponiendo que existen tres clases) repetirá este patrón de servicio. WFQ también es una disciplina de cola conservadora y, por tanto, pasará inmediatamente a la siguiente clase indicada en la secuencia de servicio cuando se encuentre con una clase de cola vacía.

WFQ se diferencia del mecanismo de turno rotatorio en que cada clase puede recibir una cantidad de servicio *diferente* en cualquier intervalo de tiempo. Específicamente, a cada clase i se le asigna un peso w_i. En WFQ, durante cualquier intervalo de tiempo en el que existan paquetes de la clase i para enviar, se garantiza que la clase i recibirá una fracción de

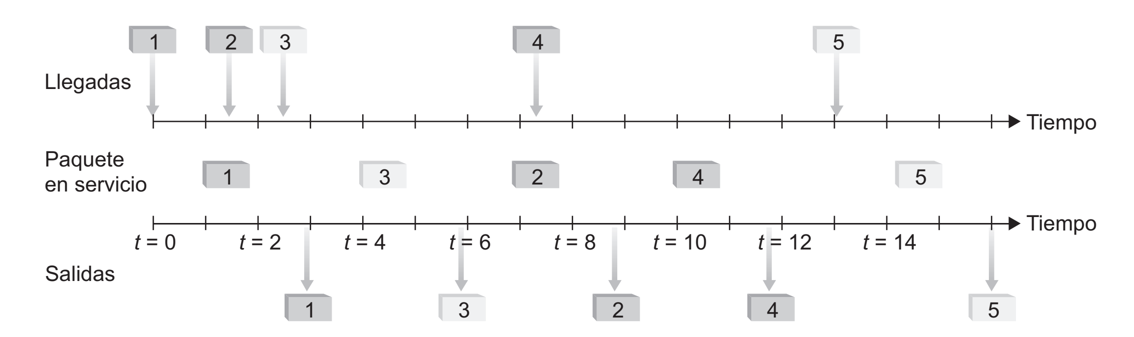


Figura 7.25 • Funcionamiento de una cola por turno rotatorio de dos clases.

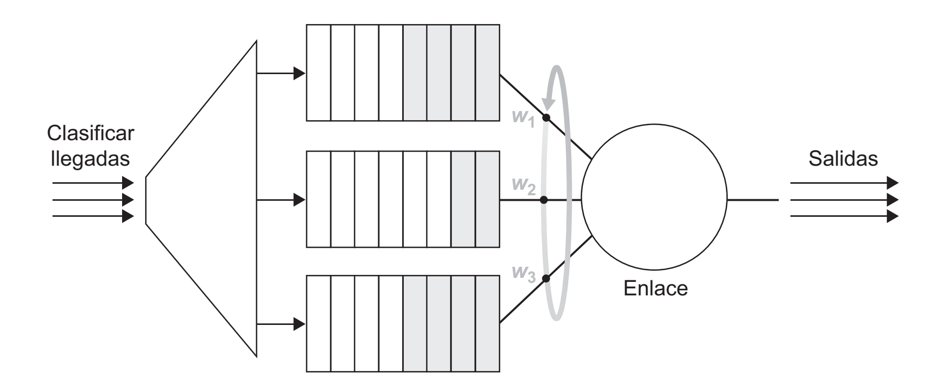


Figura 7.26 • Colas equitativas ponderadas (WFQ).

servicio igual a $w_i/(\sum w_j)$, donde el sumatorio del denominador se calcula para todas las clases que tengan paquetes para ser transmitidos. En el caso peor, incluso aunque todas las clases tengan paquetes en cola, continúa estando garantizado que la clase *i* recibirá una fracción $w_i/(\sum w_j)$ del ancho de banda. Por tanto, en el caso de un enlace con una velocidad de transmisión *R,* la clase *i* siempre alcanzará una tasa de transferencia de al menos $R \cdot w_i/(\sum w_j)$. Esta descripción de las colas WFQ que hemos proporcionado se corresponde con el caso ideal, ya que no hemos considerado el hecho de que los paquetes son unidades discretas de datos y que la transmisión de un paquete no será interrumpida por el inicio de la transmisión de otro paquete. En [Demers 1990] y [Parekh 1993] se analiza este problema. Como veremos en las siguientes secciones, WFQ desempeña un papel fundamental en las arquitecturas QoS. También está disponible en los routers actuales [Cisco QoS 2009].

Vigilancia: la cubeta con pérdidas

Uno de los principios que establecimos en la Sección 7.5.1 era que la vigilancia, es decir, la regulación de la velocidad a la que una clase o flujo puede inyectar paquetes en la red (en la exposición que sigue supondremos que la unidad de vigilancia es un flujo). Además, la vigilancia es un mecanismo QoS importante. Pero, ¿qué aspectos de la tasa de paquetes de un flujo deberían ser vigilados? Podemos identificar tres importantes criterios de vigilancia, cada uno de ellos diferente con respecto a la escala de tiempo de vigilancia del flujo de paquetes:

- *Tasa promedio.* La red puede querer limitar la tasa promedio a largo plazo (paquetes por intervalo de tiempo) a la que los paquetes de un flujo pueden ser enviados a la red. Un problema crucial en este caso es el intervalo de tiempo sobre el que se vigilará la tasa promedio. Un flujo cuya tasa promedio esté limitada a 100 paquetes por segundo está más restringido que un origen que está limitado a 6.000 paquetes por minuto, incluso aunque ambos tengan la misma tasa promedio a lo largo de un intervalo de tiempo lo suficientemente largo. Por ejemplo, esta última restricción permitiría a un flujo enviar 1.000 paquetes en un determinado intervalo de tiempo de un segundo, mientras que la primera restricción impediría este comportamiento de envíos.
- *Tasa de pico.* Mientras que la restricción de la tasa promedio limita la cantidad de tráfico que puede ser enviado a la red para un periodo de tiempo relativamente largo, la restric-

ción de la tasa de pico limita el número máximo de paquetes que pueden ser enviados en un periodo de tiempo más corto. Siguiendo con el ejemplo anterior, la red puede vigilar un flujo con una tasa promedio de 6.000 paquetes por minuto, a la vez que limita la tasa de pico del flujo a 1.500 paquetes por segundo.

- *Tamaño de la ráfaga.* La red también puede desear limitar el número máximo de paquetes (la "ráfaga" de paquetes) que pueden ser enviados a la red en un intervalo de tiempo extremadamente corto. En el límite, cuando la longitud del intervalo tiende a cero, el tamaño de la ráfaga limita el número de paquetes que pueden ser enviados a la red de forma instantánea. Incluso aunque físicamente sea imposible enviar instantáneamente varios paquetes a la red (después de todo, los enlaces tienen una velocidad de transmisión física que no puede excederse), la abstracción de un tamaño máximo de ráfaga es muy útil.

El mecanismo de goteo de la cubeta con pérdidas es una abstracción que puede utilizarse para caracterizar estos límites de vigilancia. Como se muestra en la Figura 7.27, una cubeta con pérdidas es una cubeta que puede almacenar hasta *b* fichas. Las fichas se van añadiendo a esta cubeta de la forma siguiente: las nuevas fichas que potencialmente pueden añadirse a la cubeta siempre se generan a una velocidad de *r* fichas por segundo. (Para simplificar, supondremos que la unidad de tiempo en este caso es el segundo.) Si la cubeta contiene menos de *b* fichas cuando se genera una ficha, la ficha que se acaba de generar se añade a la cubeta; en caso contrario, dicha ficha recién generada se ignora y la cubeta sigue conteniendo *b* fichas.

Veamos ahora cómo se puede utilizar una cubeta con pérdidas para vigilar o monitorizar un flujo de paquetes. Supongamos que antes de transmitir un paquete a la red es necesario eliminar una ficha de la cubeta. En este caso, si la cubeta está vacía el paquete tendrá que esperar a que haya una ficha. (Una alternativa sería eliminar el paquete, aunque esta opción no vamos a considerarla aquí.) Consideremos ahora cómo este comportamiento controla un flujo de tráfico. Puesto que como máximo puede haber *b* fichas en la cubeta, el tamaño máximo de ráfaga para un flujo controlado mediante una cubeta con pérdidas es de *b* fichas.

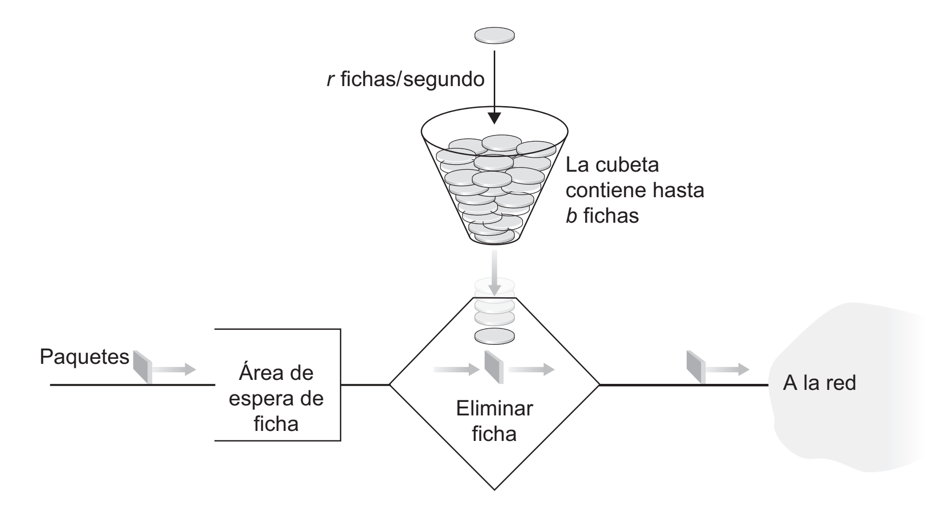


Figura 7.27 • El controlador de la cubeta con pérdidas.

Además, dado que la velocidad de generación de fichas es *r,* el número máximo de paquetes que pueden entrar en la red en *cualquier* intervalo de tiempo de longitud t es $rt + b$. Por tanto, la tasa de generación de fichas r sirve para limitar la tasa promedio a largo plazo a la que los paquetes pueden acceder a la red. También pueden utilizarse varias cubetas con pérdidas (en concreto, dos cubetas de este tipo en serie) para controlar la tasa de pico de un flujo además de la tasa promedio a largo plazo (consulte los problemas de repaso incluidos al final del capítulo).

Cubeta con pérdidas + cola WFQ = retardo máximo probable en una cola

Enseguida vamos a examinar los métodos Intserv y Diffserv que permiten proporcionar calidad de servicio en Internet. Veremos que tanto la vigilancia mediante cubetas con pérdidas como la planificación WFQ pueden desempeñar un papel importante. A continuación concluiremos esta sección considerando el enlace de salida de un router que multiplexa n flujos, estando cada uno de ellos controlado mediante una cubeta con pérdidas cuyos parámetros son b_i y r_i, $i = 1, \ldots, n$, y utilizando la disciplina de planificación WFQ. Vamos a emplear el término *flujo* para hacer referencia al conjunto de paquetes que el planificador no distingue entre sí. En la práctica, un flujo puede comprender tráfico procedente de una única conexión terminal a terminal o de una colección de conexiones de este tipo (véase la Figura 7.28).

Recuerde que con la disciplina WFQ se garantiza que cada flujo i reciba una cuota del ancho de banda del enlace igual a, como mínimo, $R \cdot w_i/(\sum w_j)$, donde R es la velocidad de transmisión del enlace en paquetes/segundo. En este caso, ¿cuál es el retardo máximo que experimentará un paquete mientras espera para recibir servicio en la cola WFQ (es decir, después de atravesar la cubeta con pérdidas)? Vamos a centrarnos en el flujo 1. Supongamos que la cubeta del flujo 1 inicialmente está llena y llega una ráfaga de b_1 paquetes al controlador de la cubeta del flujo 1. Estos paquetes extraerán todas las fichas (sin esperar) de la cubeta y pasarán al área de espera WFQ correspondiente al flujo 1. Puesto que estos b_1 paquetes reciben servicio a una tasa como mínimo de $R \cdot w_i/(\sum w_j)$ paquetes/segundo, el

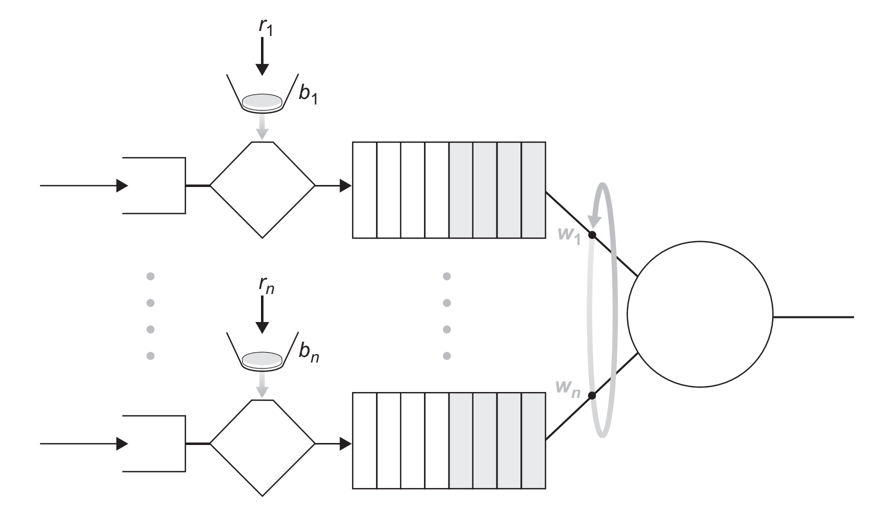


Figura 7.28 • *n* flujos multiplexados con cubetas con pérdidas y planificación WFQ.

último de estos paquetes sufrirá un retardo máximo $d_{máx}$ hasta que se complete su transmisión, donde

$$d_{máx} = \frac{b_1}{R \quad w_1/\sum w_j}$$

La fundamentación en que se basa esta fórmula es que si hay b_1 paquetes en la cola y se les está dando servicio (se les está eliminando de la cola) a una tasa de como mínimo $R \cdot w_1/(\sum w_j)$ paquetes por segundo, entonces la cantidad de tiempo transcurrido hasta que el último bit del último paquete es transmitido no puede ser mayor que $b_1/(R \cdot w_1/(\sum w_j))$. Uno de los problemas de repaso le pide que demuestre que siempre y cuando $r_1 < R \cdot w_1/(\sum w_j)$, entonces $d_{máx}$ es el retardo máximo que cualquier paquete del flujo 1 experimentará en la cola WFQ.

7.5.3 Diffserv

El objetivo de la arquitectura Diffserv de Internet [RFC 2475; Kilkki 1999] es el de proporcionar una diferenciación de servicio, es decir, la capacidad de manejar diferentes "clases" de tráfico de formas distintas dentro de Internet, y hacer esto de una manera escalable y flexible. La necesidad de la *escalabilidad* surge del hecho de que pueden existir centenares de miles de flujos simultáneos de tráfico origen-destino en un router troncal de Internet. Veremos a continuación que esta necesidad se cubre incluyendo solamente una simple funcionalidad dentro del núcleo de la red, con más operaciones de control complejas implementadas en la frontera de la red. La necesidad de la *flexibilidad* se debe al hecho de que pueden surgir nuevas clases de servicio y quedar obsoletas otras clases de servicio más antiguas. La arquitectura Diffserv es flexible en el sentido de que no define servicios ni clases de servicios específicos; en lugar de ello, Diffserv proporciona componentes funcionales, es decir, las piezas de una arquitectura de red con las que pueden construirse tales servicios. Examinemos ahora estos componentes en detalle.

Servicios diferenciados: un escenario simple

Para establecer el marco de trabajo con el fin de definir los componentes arquitectónicos del modelo de servicios diferenciados (Diffserv), comenzaremos con la sencilla red mostrada en la Figura 7.29. En esta sección vamos a describir un posible uso de los componentes de Diffserv. Como se describe en el documento RFC 2475 son posibles muchas otras variantes. Nuestro objetivo aquí es proporcionar una introducción a los aspectos clave de Diffserv, en lugar de describir el modelo arquitectónico de forma exhaustiva. Aninamos a los lectores interesados en aprender más cosas acerca de Diffserv a consultar el libro [Kilkki 1999].

La arquitectura de los servicios diferenciados consta de dos conjuntos de elementos funcionales:

- *Funciones de frontera: clasificación de paquetes y acondicionamiento del tráfico.* En la frontera de entrada de la red (es decir, en cualquier host que soporte el modelo Diffserv que genere tráfico o en el primer router compatible con Diffserv a través del cual pase el tráfico), se marcan los paquetes que llegan. Más específicamente, se asigna un cierto valor al campo servicio diferenciado (DS, *Differentiated Service*) de la cabecera del paquete. Por ejemplo, en la Figura 7.29 los paquetes que están siendo transmitidos de H1 a H3 pueden marcarse en R1, mientras que los paquetes que se envían de H2 a H4 pue-

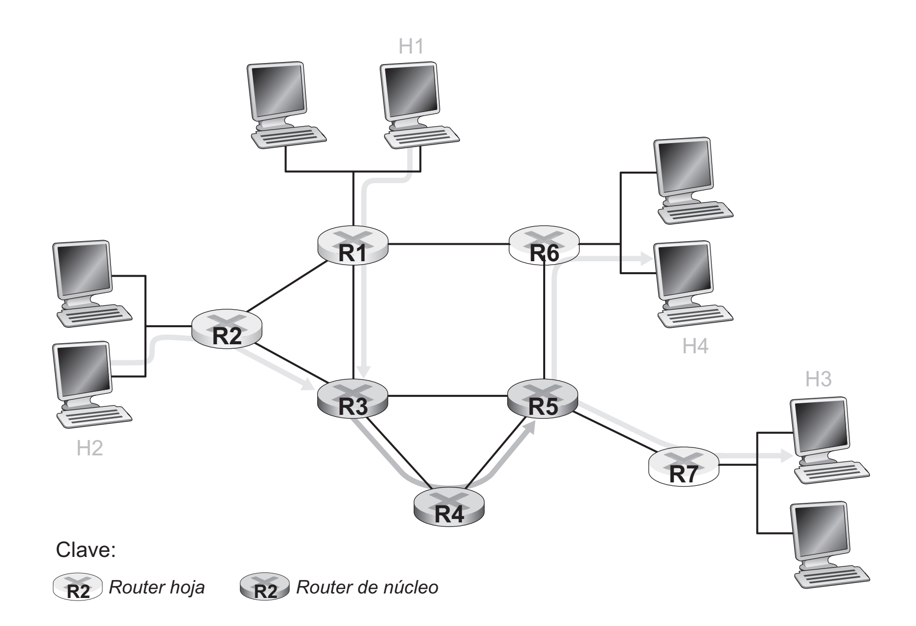


Figura 7.29 • Un ejemplo de red Diffserv simple.

den marcarse en R2. La marca que recibe un paquete identifica la clase de tráfico a la que pertenece. Las distintas clases de tráfico recibirán entonces un servicio diferente en la red principal.

- *Función del núcleo: reenvío.* Cuando un paquete marcado con DS llega a un router DS, el paquete es reenviado al siguiente salto de acuerdo con el **comportamiento por salto** asociado con dicha clase de paquete. El comportamiento por salto influye en cómo las clases de tráfico en competencia comparten los bufferes de un router y el ancho de banda del enlace. Un principio fundamental de la arquitectura Diffserv es que el comportamiento por salto de un router se basará *únicamente* en las marcas de los paquetes, es decir, en la clase de tráfico a la que pertenece el paquete. Por tanto, si los paquetes que se están enviando de H1 a H3 en la Figura 7.29 reciben la misma marca que los paquetes que están siendo enviados de H2 a H4, entonces los routers de la red tratan estos paquetes como agregados sin distinguir si los paquetes fueron originados en H1 o en H2. Por ejemplo, R3 no diferenciará entre los paquetes de H1 y H2 al reenviarlos hacia R4. Por tanto, la arquitectura de servicios diferenciados obvia la necesidad de controlar el estado del router para cada pareja individual origen-destino (una consideración importante para cumplir el requisito de escalabilidad comentado al principio de esta sección).

En este momento puede resultarnos útil comentar una analogía. En muchos actos sociales a gran escala (por ejemplo, una importante recepción pública, una sala de baile o una discoteca, un concierto o un partido de fútbol), las personas que asisten al acto reciben distintos tipos de entradas o pases: las personalidades reciben entradas VIP; los mayores de 18 años disponen de entradas de adulto (por ejemplo, en el caso de que se vayan a servir bebidas alcohólicas); los pases para estar entre bastidores en los conciertos; los pases de prensa para los periodistas, incluso un pase ordinario para las personas normales. Normalmente, estos diversos tipos de entradas o pases se distribuyen a la entrada del acto, es decir, en la frontera

que da paso al acto. Es precisamente en la frontera donde se llevan a cabo operaciones intensivas de computación, tales como abonar una entrada, comprobar que el tipo de invitación es el apropiado y comprobar que la invitación se corresponde con algún tipo de identificación. Además, puede existir un límite relativo al número de personas de un determinado tipo que pueden asistir al acto. Si existe tal límite, es posible que la gente tenga que esperar antes de entrar en el acto. Una vez que se ha accedido al acto, el pase permite que cada persona reciba un servicio diferenciado en las distintas zonas en las que tiene lugar el acto; por ejemplo, un VIP puede obtener las bebidas gratis, un mesa mejor, una comida gratuita, acceso a salas exclusivas y un servicio de atenciones. Por el contrario, una persona normal queda excluida de determinadas áreas, tendrá que abonar las bebidas y sólo recibirá un servicio básico. En ambos casos, el servicio recibido en el acto depende únicamente del tipo de pase del que disponga la persona. Además, todas las personas pertenecientes a una clase son tratadas de la misma forma.

Clasificación y acondicionamiento del tráfico de Diffserv

La Figura 7.30 proporciona una visión lógica de las funciones de clasificación y marcado en el router de frontera. En primer lugar, los paquetes que llegan al router de frontera se clasifican. El clasificador selecciona los paquetes basándose en los valores de uno o más campos de la cabecera del paquete (por ejemplo, dirección de origen, dirección de destino, puerto de origen, puerto de destino e ID de protocolo) y dirige al paquete a la función de marcado apropiada. La marca de un paquete se transporta dentro del campo DS [RFC 3260] en la cabecera del paquete IPv4 o IPv6. La definición del campo DS pretende sustituir a las definiciones anteriores del campo Tipo de servicio de IPv4 y de los campos de clase de tráfico de IPv6 que se explicaron en el Capítulo 4.

En algunos casos, un usuario final puede estar de acuerdo en limitar su velocidad de transmisión de paquetes con el fin de cumplir un **perfil declarado de tráfico**. El perfil de tráfico puede contener un límite para la tasa de pico, así como para el tamaño de ráfaga del flujo de paquetes, como hemos visto anteriormente con el mecanismo de la cubeta con pérdidas. A medida que el usuario envía paquetes a la red cumpliendo el perfil de tráfico negociado, los paquetes reciben su marca de prioridad y son reenviados a través de su ruta hasta

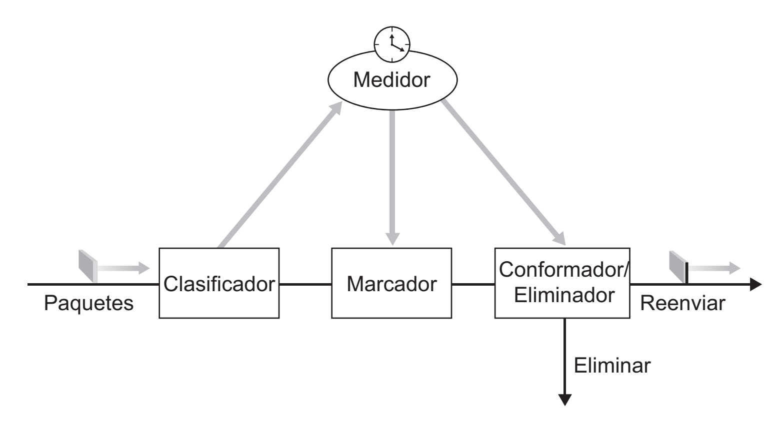


Figura 7.30 • Visión lógica de las funciones de clasificación de paquetes y acondicionamiento de tráfico en el router terminal.

alcanzar el destino. Por el contrario, si se viola el perfil de tráfico, los paquetes que no cumplen los límites impuestos por dicho perfil pueden marcarse de forma diferente, conformarse (por ejemplo, ser retardados de manera que se observe una restricción de tasa máxima), o pueden ser eliminados en la frontera de la red. El papel de la **función de medida**, mostrada en la Figura 7.30, es el de comparar el flujo de paquetes entrantes con el perfil de tráfico negociado y determinar si un paquete cumple dicho perfil negociado. La decisión real acerca de si volver a marcar de forma inmediata, reenviar, retardar o eliminar un paquete es una política que debe establecer el administrador de la red y *no* es específica de la arquitectura Diffserv.

Comportamientos por salto

Hasta el momento nos hemos centrado en las funciones de frontera de la arquitectura Diffserv. El segundo componente clave de la arquitectura de servicios diferenciados tiene que ver con el **comportamiento por salto (PHB, *Per-Hop Behavior*)** de los routers Diffserv. El PHB se define de manera críptica como "una descripción del comportamiento de reenvío observable externamente de un nodo Diffserv aplicado a un agregado de comportamiento Diffserv particular" [RFC 2475]. Profundizando un poco en esta definición, podemos ver que contiene varias consideraciones importantes:

- Un PHB puede dar lugar a que diferentes clases de tráfico reciban distintos rendimientos (es decir, diferentes comportamientos de reenvío observables externamente).
- Aunque un PHB define diferentes rendimientos (comportamientos) entre clases, no obliga al uso de ningún mecanismo concreto para conseguir estos comportamientos. Siempre y cuando se cumplan los criterios de rendimiento observables externamente, puede aplicarse cualquier mecanismo de implementación y cualquier política de asignación de buffer/ancho de banda. Por ejemplo, un PHB no requiere que se emplee una disciplina de colas de paquetes concreta (por ejemplo, una cola con prioridad, una cola WFQ o una cola FCFS) para conseguir un determinado comportamiento. El PHB es el fin y la asignación de recursos y los mecanismos de implementación son los medios.
- Las diferencias en el rendimiento deben ser observables y por tanto mensurables.

Actualmente están definidos dos PHB: un PHB de reenvío expedito (EF, *Expedited Forwarding*) [RFC 3246] y un PHB de reenvío garantizado (AF, *Assured Forwarding*) [RFC 2597].

- El PHB de **reenvío expedito** (EF) especifica que la tasa de salida de una clase de tráfico de un router tiene que ser igual o mayor que una tasa configurada. Es decir, durante cualquier intervalo de tiempo, a la clase de tráfico se le garantiza que recibirá el suficiente ancho de banda como para que la tasa de salida del tráfico sea igual o mayor que la tasa mínima configurada. Observe que el comportamiento por salto EF implica cierta forma de aislamiento entre las clases de tráfico, ya que esta garantía se implementa *independientemente* de la intensidad de tráfico de cualquier otra clase que esté llegando a un router. Por tanto, incluso aunque las restantes clases de tráfico estén inundando el router y los recursos del enlace, gran parte de estos recursos tienen que continuar estando disponibles para esa clase con el fin de garantizar que recibe su garantía de tasa mínima. Por tanto, EF proporciona una clase con la simple *abstracción* de un enlace con un ancho de banda mínimo garantizado.

- El PHB de **reenvío garantizado** (AF) es más complejo. El reenvío garantizado divide el tráfico en cuatro clases, y a cada una de las clases se le garantiza que recibirá cierta cantidad mínima de ancho de banda y de buffer. Dentro de cada clase, los paquetes vuelven a clasificarse en una de tres categorías de preferencia de eliminación. Cuando en una clase AF se produce congestión, un router puede descartar (eliminar) paquetes basándose en sus valores de preferencia de eliminación. Si desea conocer más detalles, consulte [RFC 2597]. Variando la cantidad de recursos asignados a cada clase, un ISP puede proporcionar diferentes niveles de rendimiento a las distintas clases de tráfico AF.

Retrospectiva de los servicios diferenciados

Durante los últimos 20 años han sido numerosos los intentos (la mayoría de ellos sin éxito) de introducir la calidad de servicio (QoS) en la redes de conmutación de paquetes. Los distintos intentos han fallado más por razones económicas y de legado que por cuestiones técnicas. Entre estos intentos se incluyen las redes ATM terminal a terminal y las redes TCP/IP. Veamos brevemente estos problemas en el contexto de los servicios diferenciados (los cuales estudiaremos en la siguiente sección).

Hasta el momento hemos supuesto implícitamente que Diffserv se implanta dentro de un mismo dominio administrativo. El caso más típico es aquel en el que un servicio terminal a terminal tiene que ser establecido entre varios ISP que se encuentran entre los sistemas terminales en comunicación. Para proporcionar un servicio diferenciado terminal a terminal, todos los ISP existentes entre los sistemas terminales no sólo tienen que proporcionar dicho servicio, sino que también tienen que cooperar y establecer acuerdos para ofrecer al usuario final un verdadero servicio terminal a terminal. Sin esta clase de cooperación, los ISP que venden directamente el servicio diferenciado a los clientes tendrían que decir continuamente: "Sí, sabemos que usted ha pagado un extra, pero no tenemos un acuerdo de servicio con uno de nuestros ISP de nivel superior. ¡Sentimos que se hayan producido muchos huecos en su llamada VoIP!".

Incluso dentro de un mismo dominio administrativo, Diffserv por sí solo no es suficiente para proporcionar garantías de calidad de servicio a una clase de servicio concreta. Diffserv sólo permite que diferentes clases de tráfico reciban distintos niveles de rendimiento. Si una red está extremadamente infradimensionada, incluso el tráfico con prioridad alta puede recibir un rendimiento inaceptable. Por tanto, para ser efectivos, los servicios diferenciados deben combinarse con un dimensionado apropiado de la red (véase la Sección 7.3.5). No obstante, los servicios diferenciados *pueden* sacar un mayor partido de la inversión del ISP en la capacidad de la red. Haciendo que los recursos estén disponibles para las clases de tráfico con prioridad alta (y que pagan más) cuando los necesiten (a expensas de las clases de tráfico con prioridad inferior), el ISP puede proporcionar un nivel alto de rendimiento a estas clases de prioridad alta. Cuando las clases con prioridad alta ya no necesitan estos recursos, entonces podrán utilizarlos las clases de tráfico con prioridad inferior (que presumiblemente habrán pagado menos por esta clase de servicio).

Otro problema con estos servicios avanzados es la necesidad de vigilar y, posiblemente, conformar el tráfico, lo que puede resultar complejo y costoso. También es necesario facturar los servicios de manera diferente (muy probablemente por volumen en lugar de como hacen actualmente la mayoría de los ISP, que cobran una tarifa mensual fija), otro requisito costoso para el ISP. Por último, si los servicios diferenciados estuvieran realmente implementados y la red operara con una carga sólo moderada, la mayor parte del tiempo no se per-

cibiría una diferencia entre un servicio de mejor esfuerzo y un servicio diferenciado. De hecho, actualmente, el retardo terminal a terminal normalmente se controla mediante las velocidades de acceso y los saltos de router, en lugar de mediante los retardos de cola en los routers. Imagine al infeliz cliente de un servicio diferenciado que ha pagado por un servicio premium, sólo para descubrir que el servicio de mejor esfuerzo que se está proporcionando a otros clientes tiene casi siempre el mismo rendimiento que el servicio premium.

7.6 Garantías de calidad de servicio

En la sección anterior hemos visto que el marcado y la vigilancia de paquetes, el aislamiento del tráfico y la planificación en el nivel de enlace pueden proporcionar una clase de servicio con mejor rendimiento que otra. Con ciertas disciplinas de planificación, como la planificación con prioridad, las clases de tráfico inferiores son prácticamente "invisibles" para la clase de tráfico con prioridad más alta. Con un dimensionamiento apropiado de la red, la clase de servicio de más alta prioridad puede, de hecho, conseguir tasas de pérdida de paquetes y retardos extremadamente bajos, con un rendimiento esencialmente idéntico a las redes de conmutación de circuitos. ¿Pero puede la red *garantizar* que un flujo activo perteneciente a una clase de tráfico de alta prioridad continuará recibiendo dicho servicio mientras dure el flujo, utilizando únicamente los mecanismos que hemos descrito hasta el momento? En realidad no. En esta sección veremos por qué hacen falta otros mecanismos de red y protocolos adicionales para proporcionar *garantías* de calidad de servicio.

7.6.1 Ejemplo explicativo

Volvamos a nuestro escenario de la Sección 7.5.1 y consideremos las dos aplicaciones de audio a 1 Mbps que transmiten sus paquetes a través del enlace de 1,5 Mbps, como se muestra en la Figura 7.31. La velocidad combinada de datos de los dos flujos (2 Mbps) excede la capacidad del enlace. Incluso utilizando los mecanismos de clasificación y marcado, el aislamiento de flujos y la compartición del ancho de banda no utilizado (que en realidad es igual a cero) es obvio que no podemos alcanzar nuestro objetivo. Simplemente, no existe el suficiente ancho de banda como para satisfacer las necesidades de ambas aplicaciones al mismo tiempo. Si las dos aplicaciones comparten equitativamente el ancho de banda, cada una de ellas sólo podrá emplear 0,75 Mbps. Examinando las cosas desde otro punto de vista, cada aplicación perdería el 25 por ciento de sus paquetes transmitidos. Ésta es una calidad de servicio tan inaceptablemente baja que ambas aplicaciones de audio serán completamente inutilizables; de hecho, ni siquiera merece la pena transmitir ningún paquete de audio.

Dado que no se puede satisfacer simultáneamente a las dos aplicaciones de la Figura 7.31, ¿qué debería hacer la red? Permitir que ambas continúen con una QoS inaceptable implica desperdiciar los recursos de la red en una serie de flujos de aplicación que, en último extremo, no tienen ninguna utilidad para el usuario final. La respuesta es bastante simple: habrá que bloquear uno de los flujos de aplicación (es decir, denegarle el acceso a la red) mientras que se permite al otro continuar utilizando el 1 Mbps completo que la aplicación necesita. La red telefónica sería un ejemplo de red en la que se efectúa ese tipo de bloqueo de llamadas: si no se pueden asignar a la llamada los recursos requeridos (un circuito terminal a terminal en el caso de la red telefónica), se bloquea la llamada (se la impide cursarse a través de la red) y se devuelve una señal de ocupado al usuario. En nuestro ejemplo, no se

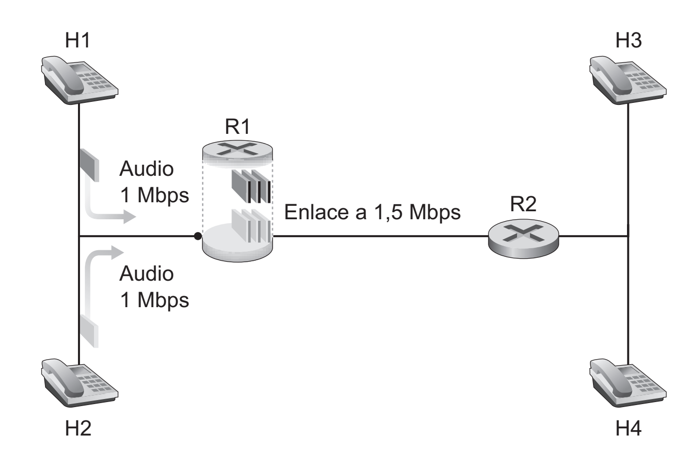


Figura 7.31 • Dos aplicaciones de audio competidoras sobrecargan el enlace de R1a R2.

gana nada permitiendo que un flujo entre en la red si no va a recibir una QoS suficiente como para poder considerarlo utilizable. De hecho, existe un *coste* asociado a la admisión de un flujo que no vaya a recibir su QoS necesaria, ya que se estarán empleando recursos de la red para dar soporte a un flujo que no tiene ninguna utilidad para el usuario final.

Admitiendo o bloqueando explícitamente los flujos, basándose en sus requisitos de recursos y en los requisitos de los flujos ya admitidos, la red puede garantizar que los flujos admitidos podrán recibir la QoS solicitada. Junto a la necesidad de proporcionar una QoS garantizada a un cierto flujo está implícita la necesidad de que ese flujo declare sus requisitos de QoS. Este proceso de hacer que un flujo declare sus requisitos de QoS y que luego la red acepte el flujo (con la QoS requerida) o le bloquee se denomina proceso de **admisión de llamada**. Éste es, por tanto, el cuarto de los principios fundamentales (de los otros tres hemos hablado en la Sección 7.5.1) de los mecanismos necesarios para proporcionar calidad de servicio (QoS).

> **Principio 4:** si los recursos suficientes no siempre van a estar disponibles y es necesario *garantizar* la calidad de servicio, se necesita un proceso de admisión de llamadas en el que los flujos declaren sus requisitos de QoS y o bien los admita en la red (con la QoS requerida) o bloquee (si la red no puede proporcionar la QoS requerida).

7.6.2 Reserva de recursos, admisión de llamadas, establecimiento de llamadas

Nuestro ejemplo introductorio resalta la necesidad de diversos nuevos mecanismos y protocolos de red, para el caso en que haya que garantizar a una llamada (a un flujo terminal a terminal) una determinada calidad de servicio una vez que la llamada se ha establecido:

- *Reserva de recursos.* La única forma de *garantizar* que una llamada dispondrá de los recursos (ancho de banda de enlace, buffers) necesarios para obtener su calidad de servicio deseada consiste en asignar explícitamente dichos recursos a esa llamada; a este proceso se le conoce en la jerga del mundo de las redes con el nombre de **reserva de recursos**. Una vez reservados los recursos, la llamada podrá, mientras dure, acceder bajo

demanda a dichos recursos, independientemente de las demandas de todas las restantes llamadas. Si una llamada reserva y recibe una garantía de x Mbps de ancho de banda de enlace y nunca transmite a una velocidad superior a x podrá disfrutar de unas comunicaciones sin pérdidas y sin retardos.

- *Admisión de llamadas.* Si hay que reservar recursos, entonces la red tiene que disponer de un mecanismo para que las llamadas puedan solicitar y reservar los recursos; este proceso se conoce con el nombre de proceso de admisión de llamadas. Dado que los recursos no son infinitos, cuando una llamada realiza una solicitud de admisión de llamada esa admisión será denegada (es decir, la llamada será bloqueada) si no están disponibles los recursos solicitados. Este tipo de admisión de llamadas es realizado por las redes telefónicas, en las que solicitamos los recursos en el momento de marcar un número. Si están disponibles los circuitos (particiones TDMA) necesarios para completar la llamada, se asignarán los circuitos y la llamada podrá completarse. Si los circuitos no están disponibles, entonces se bloqueará la llamada y recibiremos la señal de ocupado. Una llamada bloqueada puede volver a intentar que la admitan en la red, pero no se le permitirá enviar tráfico hacia la red hasta que haya completado el proceso de admisión de llamada.

  Por supuesto, al igual que el gestor del restaurante de la Sección 1.3.1 no debería aceptar reservas para más mesas que las que el restaurante tiene, un router que realiza asignaciones del ancho de banda del enlace no debe asignar más ancho de banda del que esté disponible en dicho enlace. Normalmente una llamada sólo podrá reservar una fracción del ancho de banda del enlace, por lo que el router puede asignar ancho de banda del enlace a más de una llamada. Sin embargo, la suma del ancho de banda asignado a todas las llamadas debe ser inferior a la capacidad del enlace.

- *Señalización del establecimiento de llamada.* El proceso de admisión de llamadas descrito más arriba requiere que las llamadas sean capaces de reservar los recursos suficientes en cada uno de los routers que formen parte de la ruta entre el origen y el destino, con el fin de asegurarse de satisfacer sus requisitos de QoS terminal a terminal. Cada router deberá determinar los recursos locales requeridos por la sesión, tener en cuenta la cantidad de recursos que ya han sido comprometidos con otras sesiones activas y determinar si dispone de los suficientes recursos como para satisfacer los requisitos de QoS por salto que esa sesión tiene en dicho router, sin violar las garantías de QoS locales que ya se hayan concedido a otras sesiones ya admitidas. Hace falta un protocolo de señalización para coordinar estas diversas actividades: la asignación de recursos locales en cada salto, así como la decisión global terminal a terminal de si la llamada ha sido o no capaz de reservar los recursos suficientes en cada uno de los routers de la ruta terminal a terminal. Éste es el trabajo del **protocolo de establecimiento de llamada**.

En la Figura 7.32 se describe el proceso de establecimiento de llamada. Consideremos ahora con más detalle los pasos implicados en un proceso de admisión de llamada:

1. *Caracterización del tráfico y especificación de la calidad de servicio deseada.* Para que un router pueda determinar si sus recursos son suficientes o no para satisfacer los requisitos QoS de una llamada, dicha llamada debe declarar en primer lugar sus requisitos QoS, así como caracterizar el tráfico que va a enviar a la red y para el que está solicitando una garantía de QoS. En la arquitectura Intserv de Internet, la especificación Rspec (donde la letra R hace referencia al término reserva) [RFC 2215] define la QoS

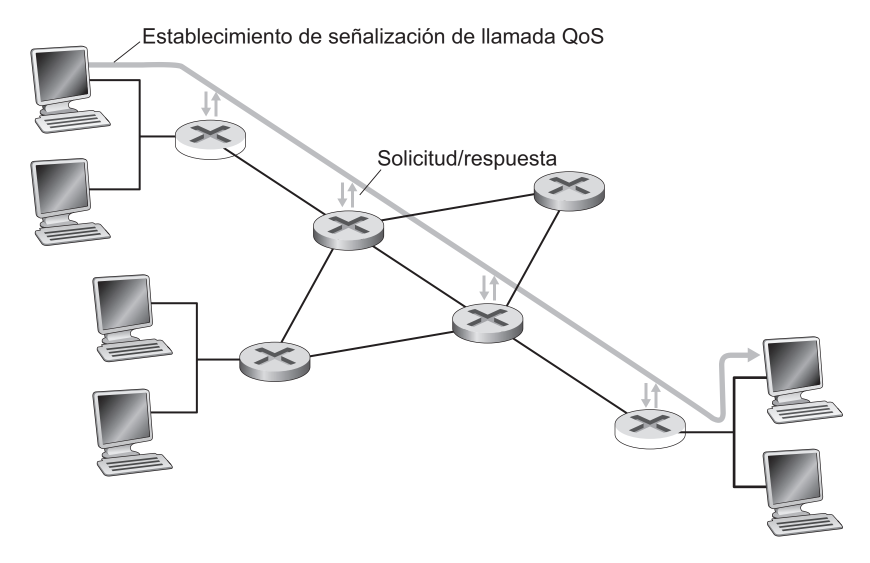


Figura 7.32 • Proceso de establecimiento de llamada.

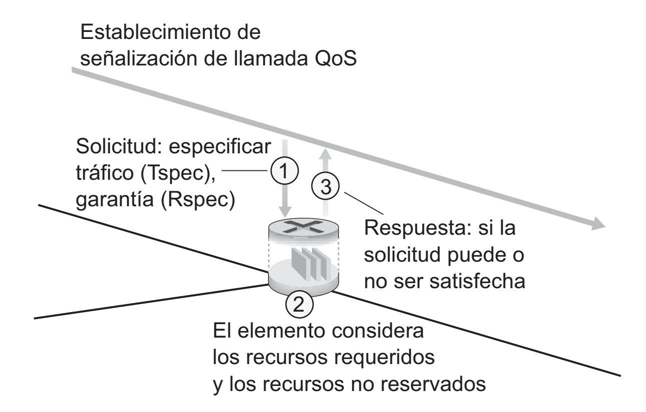


Figura 7.33 • Comportamiento de cada elemento con respecto a la llamada.

específica que una llamada está solicitando; la especificación denomina Tspec (donde la letra T hace referencia al término tráfico) [RFC 2210] caracteriza el tráfico que el emisor va a enviar hacia la red o que el receptor va a recibir de la red, respectivamente. La forma específica de Rspec y Tspec variará dependiendo del servicio solicitado, como se explica más adelante. En las redes ATM, los elementos de información referentes a la descripción del tráfico de usuario y al parámetro QoS transportan información con propósitos similares a las especificaciones Tspec y Rspec, respectivamente.

2. *Señalización para el establecimiento de llamada.* Es necesario transportar hasta los routers en los que se van a reservar los recursos para la llamada un descriptor de tráfico de la llamada y una solicitud de QoS. En Internet, dentro de la arquitectura Intserv, se utiliza para este propósito el protocolo RSVP [RFC 2210]. En las redes ATM, el protocolo Q2931b transporta esta información entre los conmutadores de la red ATM y un punto terminal.



EL PRINCIPIO DEL ESTADO FRÁGIL

RSVP se utiliza para instalar el estado (reservas de ancho de banda) en los routers y se dice de él que es un protocolo de estado frágil (*soft-state*). En términos generales, asociamos el término *estado frágil* con aquellas técnicas de señalización en las que el estado instalado caduca (y es eliminado), a menos que sea refrescado periódicamente mediante la recepción de un mensaje de señalización (normalmente enviado por la entidad que instaló inicialmente el estado) que indique que el estado debe permanecer instalado. Puesto que un estado no refrescado terminará por caducar, la señalización de estado frágil no requiere ni una eliminación explícita del estado, ni un procedimiento para eliminar los estados huérfanos en caso de que fallara el instalador de estados. De forma similar, puesto que la instalación del estado y los mensajes de refresco serán seguidos por sucesivos mensajes de refresco periódicos, no hace falta que la señalización sea fiable. El término *estado frágil* fue acuñado por Clark [Clark 1988], quien describió la noción de mensajes periódicos de refresco del estado enviados por un sistema terminal y sugirió que, con dichos mensajes de refresco, el estado podría perderse en caso de fallo y luego ser restaurado automáticamente mediante los mensajes de refresco subsiguientes, todo ello de forma transparente para el sistema terminal y sin invocar ningún procedimiento explícito de recuperación de fallos:

> ". . . la información de estado no sería crítica a la hora de mantener el tipo deseado de servicio asociado con el flujo. En lugar de ello, ese tipo de servicio sería impuesto por los puntos terminales, que enviarían mensajes periódicamente para garantizar que se asocie con el flujo el tipo de servicio apropiado. De esta forma, la información de estado asociada con el flujo podría perderse como consecuencia de fallo, sin que ello provocara una interrupción permanente de las características de servicio que se estén utilizando. Yo llamo a este concepto "estado frágil" y puede que dicho concepto nos permita conseguir nuestros objetivos principales de supervivencia y flexibilidad. . ."

Simplificando, la esencia de una técnica basada en estado frágil consiste en que el instalador del estado utilice un procedimiento periódico de instalación/refresco del estado basado en el mejor esfuerzo y otro procedimiento de eliminación del estado por caducidad en el dispositivo donde ese estado se almacene. Las técnicas de estado frágil han sido adoptadas en numerosos protocolos, incluyendo RSVP, PIM (Sección 4.7), SIP (Sección 7.4.3) e IGMP (Sección 4.7), así como en las tablas de reenvío de puentes transparentes (Sección 5.6).

La señalización de *estado firme (hard-state)* utiliza la técnica inversa a la del estado frágil: el estado instalado continuará instalado hasta que sea explícitamente eliminado por el receptor de un mensaje de eliminación del estado enviado por el instalador del estado. Puesto que el estado continúa instalado hasta que sea explícitamente eliminado, la señalización de estado firme requiere un mecanismo para eliminar los estados huérfanos que hayan quedado después de que falle el instalador de estado o después de que éste haya terminado su ejecución sin eliminar algún estado. De forma similar, dado que la instalación y la eliminación del estado sólo se llevan a cabo una vez (sin refrescos ni caducidad del estado), es importante para el instalador del estado conocer cuándo ha sido instalado o eliminado dicho estado. Como consecuencia, con las técnicas de estado firme suelen asociarse protocolos de señalización fiables (en lugar de basados en el mejor esfuerzo). Por tanto, y por simplificar nuestro análisis, la esencia de una técnica de estado firme consiste en la instalación y eliminación fiables y explícitas de la información de estado. Las técnicas de estado firme

se han adoptado en protocolos tales como ST-II [Partridge 1992, RFC 1190] y Q.2931 [ITU-T Q.2931 1994].

RSVP ha proporcionado desde su concepción un mecanismo para la eliminación explícita (aunque opcional) de las reservas.

La señalización fiable basada en mensajes de reconocimiento ACK fue introducida como extensión de RSVP en [RFC 2961] y también fue sugerida en [Pan 1997]. RSVP ha adoptado opcionalmente, por tanto, algunos elementos de las técnicas de señalización de estado firme. Consulte [Ji 2003] para ver un análisis y una comparativa de los protocolos de estado frágil y de estado firme.

---

3. *Admisión de llamadas en cada elemento.* Una vez que un router recibe la especificación de tráfico y el parámetro QoS, debe determinar si puede o no admitir la llamada. Esta decisión sobre la admisión de la llamada dependerá de la especificación de tráfico, del tipo de servicio solicitado y de los compromisos de recursos existentes que el router haya aceptado para las llamadas que ya están activas. Recuerde que hemos visto, por ejemplo en la Sección 7.5.3, cómo puede utilizarse la combinación de un origen controlado mediante una cubeta con pérdidas y WFQ para determinar el retardo máximo de puesta en cola en dicho origen. En la Figura 7.33 se ilustra el proceso de admisión de una llamada en un cierto elemento.

Consulte [Breslau 2000; Roberts 2004] para ver más detalles acerca del proceso de admisión y establecimiento de llamadas.

7.6.3 QoS garantizada en Internet: Intserv y RSVP

La arquitectura de servicios integrados (**Intserv**) es un marco de trabajo desarrollado por el IETF para proporcionar garantías QoS individualizadas a cada sesión de aplicación individual en Internet. La especificación de servicio garantizado en Intserv, definida en [RFC 2212], proporciona cotas firmes (matemáticamente demostrables) para los retardos de puesta en cola que un paquete puede experimentar en un router. Aunque los detalles subyacentes al servicio garantizado son bastante complicados, la idea básica es, en realidad, muy simple. En una primera aproximación, se proporciona una caracterización del tráfico de un cierto origen mediante un mecanismo de cubeta con pérdidas (véase la Sección 7.5.2) con parámetros (r,b) y el servicio solicitado se caracteriza mediante la velocidad de transmisión, R, a la que se enviarán los paquetes. En esencia, una llamada que esté solicitando un servicio garantizado estará pidiendo que se garantice a los bits de sus paquetes una velocidad de reenvío de R bits/segundo. Dado que el tráfico se especifica mediante una caracterización basada en el mecanismo de la cubeta con pérdidas y dado que se está solicitando una velocidad garantizada R, también es posible acotar el retardo máximo de puesta en cola en el router. Recuerde que, con una caracterización del tráfico mediante la cubeta con pérdidas, la cantidad de tráfico (en bits) generada en cualquier intervalo de duración t está acotada por la expresión $rt + b$. Recuerde también de la Sección 7.5.2 que cuando se conecta un origen con cubeta con pérdidas a una cola que garantiza que se dará servicio al tráfico a una velocidad no inferior a R bits por segundo, el retardo máximo de puesta en cola experimentado por cualquier paquete estará acotado por la expresión b/R, siempre y cuando R sea mayor que r. También se ha definido una segunda forma de garantía de servicio Intserv, conocida como

servicio de carga controlada, que especifica que una llamada recibirá una "calidad de servicio que se aproximará grandemente a la QoS que ese mismo flujo recibiría de un elemento de red no cargado" [RFC 2211].

El Protocolo de reserva de recursos (RSVP, *Resource ReSerVation Protocol*) [RFC 2205; Zhang 1993] es un protocolo de señalización de Internet que podría utilizarse para llevar a cabo la señalización de establecimiento de llamada que Intserv necesita. RSVP también se ha utilizado junto con Diffserv para coordinar funciones de servicios diferenciados entre múltiples redes, y también ha sido ampliada y utilizada como protocolo de señalización en otras circunstancias, entre las que cabe destacar la variante RSVP-TE [RFC 3209] para señalización MPLS, como se ha visto en la Sección 5.8.2.

En un contexto Intserv, el protocolo RSVP permite a las aplicaciones reservar ancho de banda para sus flujos de datos. El protocolo es empleado por un host, por cuenta de un flujo de datos de aplicación, con el fin de solicitar una cantidad específica de ancho de banda de la red. RSVP también es utilizado por los routers para reenviar las solicitudes de reserva de ancho de banda. Para implementar RSVP es necesario que el software RSVP esté presente en los receptores, los emisores y en los routers situados a lo largo de la ruta terminal a terminal mostrada en la Figura 7.32. Las características principales de RSVP son:

- Proporciona **reservas de ancho de banda en árboles de multidifusión**, gestionándose la unidifusión como un caso degenerado de multidifusión. Esto es particularmente importante para las aplicaciones multimedia, tales como los flujos de televisión sobre IP, en los que muchos receptores pueden querer recibir el mismo tráfico multimedia que se está enviando desde un único origen.
- Es **orientado al receptor**; es decir, el receptor de un flujo de datos se encarga de iniciar y mantener la reserva de recursos utilizada para dicho flujo. Esta visión innovadora, centrada en el receptor de RSVP, asigna a los receptores el control del tráfico que reciben, permitiendo por ejemplo que distintos receptores reciban y visualicen una multidifusión multimedia a diferentes resoluciones. Esto contrasta bastante con la visión centrada en el emisor de la señalización adoptada en el protocolo Q2931b de ATM.

El estándar RSVP [RFC 2205] no especifica *cómo* la red debe proporcionar el ancho de banda reservado a los flujos de datos. Se trata meramente de un protocolo que permite a las aplicaciones reservar el ancho de banda de enlace necesario. Una vez hechas las reservas, es tarea de los routers de Internet proporcionar ese ancho de banda reservado a los distintos flujos de datos. Estas asignaciones probablemente se hagan utilizando los mecanismos de vigilancia y planificación (cubeta con pérdidas, planificación con prioridades, colas equitativas ponderadas) analizados en la Sección 7.5. Para obtener más información acerca de RSVP, consulte [RFC 2205; Zhang 1993] y los materiales electrónicos en línea adicionales asociados con este libro.

7.7 Resumen

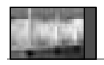
El campo de las redes multimedia constituye uno de los desarrollos más atractivos (y que todavía no han terminado de materializarse) en la red Internet actual. Las personas están en todo el mundo pasando cada vez menos tiempo delante de su radio o de su televisión y recurriendo en su lugar a Internet para acceder a transmisiones de audio y de vídeo, tanto en vivo

como pregrabadas. A medida que vaya habiendo más y más hogares con acceso de alta velocidad, esta tendencia no hará sino acelerarse: los telespectadores típicos de todo el mundo accederán a sus programas de vídeo favoritos a través de Internet en lugar de a través de los canales tradicionales de señales de televisión. Además de para la distribución de audio y de vídeo, Internet también se está empleando para transportar llamadas telefónicas. De hecho, a lo largo de la próxima década Internet puede llegar a hacer que el sistema telefónico tradicional de conmutación de circuitos quede obsoleto en muchos países. Internet no sólo proporcionará servicios telefónicos por menos dinero, sino que también facilitará numerosos servicios de valor añadido, como videoconferencia, servicios de directorio en línea, servicios de mensajería vocal e integración web.

En la Sección 7.1 hemos clasificado las aplicaciones multimedia en tres categorías diferentes: flujos de audio y vídeo almacenado, transmisión uno-a-muchos de audio y vídeo en tiempo real y audio y vídeo interactivo en tiempo real. Hemos recalcado que las aplicaciones multimedia son sensibles al retardo y tolerantes a las pérdidas, características muy distintas a las de las aplicaciones con contenido estático, que son tolerantes al retardo e intolerantes a las pérdidas. También hemos explicado algunas de las dificultades que el servicio actual de entrega de mejor esfuerzo de Internet plantea a las aplicaciones multimedia. Hemos repasado diversas propuestas que se han hecho para resolver esos problemas planteados, incluyendo la propuesta de mejorar simplemente la infraestructura de red existente (añadiendo más ancho de banda, más cachés de red y más nodos CDN e implantando tecnologías de multidifusión), la propuesta de añadir funcionalidad a Internet de manera que las aplicaciones puedan reservar recursos terminal a terminal de modo que la red pueda satisfacer las reservas realizadas y, finalmente, la propuesta de introducir clases de servicio para permitir diferenciar los distintos servicios.

En las Secciones 7.2 a 7.4 hemos examinado las arquitecturas y mecanismos para la transmisión de información multimedia en una red con un servicio de entrega de mejor esfuerzo. En la Sección 7.2 hemos repasado diversas arquitecturas para la transmisión de flujos de audio y vídeo almacenado. Hemos analizado la interacción con los usuarios (las órdenes de pausa/reanudación, reposicionamiento y avance rápido visual) y hemos hecho una introducción a RTSP, un protocolo que proporciona interacción cliente-servidor para las aplicaciones de flujos multimedia. En la Sección 7.3 hemos examinado cómo pueden diseñarse aplicaciones interactivas en tiempo real para ejecutarse sobre una red con servicio de entrega de mejor esfuerzo. Allí vimos cómo pueden aliviarse enormemente los efectos de las fluctuaciones inducidas por la red utilizando una combinación de bufferes de cliente, números de secuencia de los paquetes y marcas de tiempo. También hemos explicado cómo una CDN facilita la transmisión de flujos multimedia almacenados, al transmitir proactivamente la información multimedia almacenada a servidores CDN situados “cerca” de los puntos terminales de usuario.

En la Sección 7.5 hemos visto cómo pueden utilizarse diversos mecanismos de red (disciplinas de planificación de nivel de enlace y mecanismos de vigilancia del tráfico) para proporcionar un servicio diferenciado entre distintas clases de tráfico. Finalmente, en la Sección 7.6 hemos analizado cómo puede una red proporcionar *garantías* de calidad de servicio a las llamadas admitidas por la red. Allí vimos que hacen falta otros nuevos protocolos y mecanismos de red adicionales, incluyendo los de reserva de recursos, admisión de llamadas y señalización de llamadas. Juntos, estos nuevos elementos de red hacen que las redes del mañana, capaces de proporcionar garantías QoS, vayan a ser muy diferentes (y bastante más complejas) de la Internet actual basada en un servicio de entrega de mejor esfuerzo.



Problemas y cuestiones de repaso

Capítulo 7 Cuestiones de repaso

SECCIONES 7.1–7.2

R1. ¿Qué quiere decir interactividad en la transmisión de flujos de audio/vídeo almacenado? ¿Y en la transmisión de audio/vídeo interactivo en tiempo real?

R2. Se han expuesto tres campos de mejora de Internet para dar un mejor soporte a las aplicaciones multimedia. Resuma brevemente los puntos de vista de cada campo. ¿A qué campo pertenece usted?

R3. ¿Cuáles son algunas de las tasas de compresión típicas (relación entre el número de bits de un objeto no comprimido y el número de bits de la versión comprimida de dicho objeto) en las aplicaciones de audio e imágenes? ¿Cuáles son las técnicas de compresión examinadas en la Sección 7.1?

R4. Las Figuras 7.1 y 7.2 presentan dos esquemas de transmisión de flujos multimedia almacenados. ¿Cuáles son las ventajas y desventajas de cada uno de ellos?

SECCIONES 7.3–7.4

R5. ¿Cuál es la diferencia entre el retardo terminal a terminal y la fluctuación de paquetes? ¿Cuáles son las causas de la fluctuación de paquetes?

R6. ¿Por qué un paquete que se recibe después de su instante de reproducción planificado se considera un paquete perdido?

R7. En la Sección 7.3 se han descrito dos esquemas FEC. Resúmalos brevemente. Ambos esquemas incrementan la velocidad de transmisión del flujo mediante la adición de más sobrecarga. ¿El intercalado aumenta también la velocidad de transmisión?

R8. ¿Cuál es el papel de DNS en una red CDN? ¿Tiene que modificarse el sistema DNS para dar soporte a una CDN? ¿Qué información tiene que proporcionar una CDN al sistema DNS, si es que tiene que proporcionar alguna?

R9. ¿Qué información se necesita para dimensionar una red de modo que se consiga una determinada calidad de servicio?

R10. ¿Cuántos flujos RTP distintos de sesiones diferentes puede identificar un receptor? ¿Cómo se identifican los diferentes flujos dentro de la misma sesión? ¿Cómo se distinguen los paquetes RTP y RTPC (como parte de la misma sesión)?

R11. En la Sección 7.4 se han descrito tres tipos de paquetes RTCP. Resuma brevemente la información contenida en cada uno de estos tipos de paquetes.

R12. ¿Cuál es el papel de un registrador SIP? ¿En qué se diferencia el papel de un registrador SIP del de un agente propio en IP móvil?

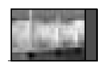
SECCIONES 7.5–7.6

R13. En la Sección 7.5 hemos visto las colas con prioridad no expropiativa. ¿En qué consistirían las colas con prioridad expropiativa? ¿Tienen sentido las colas con prioridad expropiativa en las redes de computadoras?

R14. Proporcione un ejemplo de disciplina de planificación de trabajo que *no* sea conservadora.

R15. Proporcione un ejemplo de las colas que haya en su vida cotidiana y que se correspondan con los esquemas de las colas FIFO, con prioridad, por turno rotatorio y WFQ.

R16. ¿Cuáles son algunas de las dificultades asociadas con el modelo Intserv y la reserva de recursos por parte de los flujos?



Problemas

P1. Navegue por la Web y localice dos sitios en los que haya disponibles flujos de audio y/o vídeo almacenado. Para cada sitio, utilice Wireshark para determinar:

a. Si se utilizan metarchivos.

b. Si el audio/vídeo se envía sobre UDP o sobre TCP.

c. Si se utiliza RTP.

d. Si se utiliza RTSP.

P2. Considere el buffer de cliente mostrado en la Figura 7.3. Suponga que el sistema de transmisión de flujos utiliza la tercera opción; es decir, el servidor envía los datos multimedia al socket tan rápido como es posible. Suponga que el ancho de banda TCP disponible es >> d la mayor parte del tiempo. Suponga también que el buffer de cliente sólo almacena la tercera parte de los datos. Describa cómo $x(t)$ y el contenido del buffer de cliente variarán a lo largo del tiempo.

P3. ¿Es lo mismo el buffer de recepción TCP y el buffer de cliente del reproductor multimedia? Si no son lo mismo, ¿cómo interactúan?

P4. En el ejemplo de telefonía por Internet de la Sección 7.3, sea h el número total de bytes de cabecera añadido a cada fragmento, incluyendo las cabeceras UDP e IP.

a. Suponiendo que se envía un datagrama IP cada 20 milisegundos, determine la velocidad de transmisión en bits por segundo para los datagramas generados por un lado de esta aplicación.

b. ¿Cuál es el valor típico de h cuando se utiliza RTP?

P5. Considere el procedimiento descrito en la Sección 7.3 para estimar el retardo promedio d_i. Suponga que u = 0,1. Sea $r_1 - t_1$ el retardo de la muestra más reciente, sea $r_2 - t_2$ el siguiente retardo de la muestra más reciente, etc.

a. Para una aplicación de audio dada suponga que han llegado cuatro paquetes al receptor con los retardos de muestra $r_4 - t_4$, $r_3 - t_3$, $r_2 - t_2$ y $r_1 - t_1$. Exprese la estimación del retardo d en función de las cuatro muestras.

b. Generalice la fórmula para n retardos de muestra.

c. Para la fórmula del apartado (b), obtenga la fórmula resultante cuando n tiende a infinito. Comente por qué este procedimiento para obtener la media se denomina media móvil exponencial.

P6. Repita los apartados (a) y (b) del problema anterior para obtener la estimación de la desviación media del retardo.

P7. En el ejemplo de telefonía por Internet de la Sección 7.3, hemos presentado un procedimiento en línea (media móvil exponencial) para estimar el retardo. En este problema vamos a examinar un procedimiento alternativo. Sea t_i la marca de tiempo del paquete *i-ésimo* recibido; sea r_i el instante en el que el paquete *i-ésimo* es recibido. Sea d_n nuestra estimación del retardo medio después de recibir el paquete *n-ésimo*. Después de recibir el primer paquete, establecemos que la estimación del retardo es igual a $d_1 = r_1 - t_1$.

a. Suponga que deseamos que $d_n = (r_1 - t_1 + r_2 - t_2 + \ldots + r_n - t_n)/n$ para todo n. Proporcione una fórmula recursiva para d_n en función de d_{n-1}, r_n y t_n.

b. Describa por qué en la telefonía por Internet la estimación del retardo descrita en la Sección 7.3 es más apropiada que la estimación dada en el apartado (a).

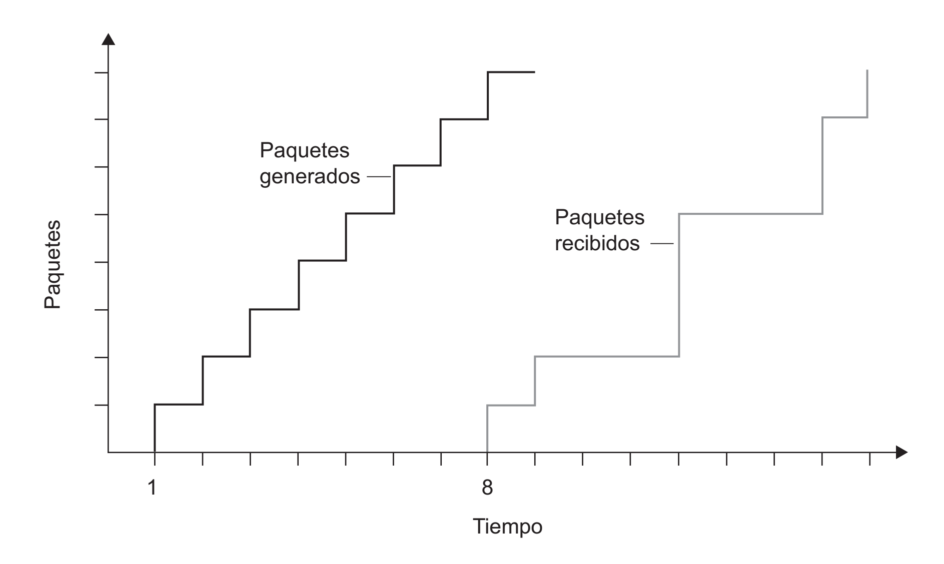
P8. Compare el procedimiento descrito en la Sección 7.3 para estimar el retardo medio con el procedimiento dado en la Sección 3.5 para estimar el tiempo de ida y vuelta. ¿Qué tienen en común ambos procedimientos? ¿En qué se diferencian?

P9. Considere la estrategia de reproducción adaptativa expuesta en la Sección 7.3.

a. ¿Cómo pueden tener dos paquetes recibidos en el destino sucesivamente marcas de tiempo que difieren en más de 20 milisegundos si ambos paquetes pertenecen al mismo periodo de conversación?

b. ¿Cómo puede utilizar el receptor los números de secuencia para determinar si se trata del primer paquete de un periodo de conversación? Sea específico.

P10. Considere la siguiente figura (que es similar a la Figura 7.5). Un emisor comienza a enviar audio empaquetado periódicamente en $t = 1$. El primer paquete llega al receptor en $t = 8$.



a. ¿Cuáles son los retardos (del emisor al receptor, ignorando cualquier retardo de reproducción) de los paquetes 2 a 8? Observe que cada segmento de línea vertical y horizontal de la figura tiene una longitud de 1, 2 o 3 unidades de tiempo.

b. Si la reproducción del audio se inicia tan pronto como llega el primer paquete al receptor en t = 8, ¿cuáles de los ocho primeros paquetes enviados no llegarán a tiempo para la reproducción?

c. Si la reproducción del audio comienza en t = 9, ¿cuáles de los ocho primeros paquetes enviados no llegarán a tiempo para la reproducción?

d. ¿Cuál es el retardo mínimo de reproducción en el receptor que hace que los ocho primeros paquetes lleguen a tiempo para la reproducción?

P11. Considere de nuevo la figura del Problema P10, que muestra los tiempos de transmisión y de recepción de los paquetes.

a. Calcule el retardo estimado para los paquetes 2 hasta 8 utilizando la fórmula para d_i de la Sección 7.3.2. Utilice un valor de $u = 0,1$.

b. Calcule la desviación estimada del retardo respecto del promedio estimado para los paquetes 2 a 8 utilizando la fórmula para v_i de la Sección 7.3.2. Utilice un valor de $u = 0,1$.

P12. Recuerde los dos esquemas FEC para la aplicación de telefonía por Internet descritos en la Sección 7.3. Suponga que el primer esquema genera un fragmento redundante por cada cuatro fragmentos originales. Suponga que el segundo esquema utiliza una codificación con una tasa de bit baja cuya velocidad de transmisión es el 25 por ciento de la velocidad de transmisión del flujo nominal.

a. ¿Cuánto ancho de banda adicional requiere cada esquema? ¿Cuánto retardo de reproducción añade cada esquema?

b. ¿Cómo funciona cada uno de los dos esquemas cuando se pierde el primer paquete de cada grupo de cinco paquetes? ¿Qué esquema proporcionará una mejor calidad de audio?

c. ¿Cómo funciona cada uno de los dos esquemas cuando se pierde el primer paquete de cada grupo de dos paquetes? ¿Qué esquema proporcionará una mejor calidad de audio?

P13. Puesto que una CDN no incrementa la cantidad de capacidad de enlace en una red (suponiendo que la CDN utiliza los enlaces existentes para distribuir su contenido entre los nodos CDN), ¿cómo mejora una CDN el rendimiento visto por los hosts? Proporcione un ejemplo.

P14. ¿Es posible para una CDN proporcionar un peor rendimiento a un host que ha solicitado un objeto multimedia, que cuando el host solicita el objeto al servidor origen distante? Explique su respuesta.

P15. ¿Cómo se calcula la fluctuación temporal entre llegadas en el informe de recepción de RTCP? (*Sugerencia*: lea el documento RFC dedicado a RTP.)

P16. a. Suponga que hemos enviado a Internet dos datagramas IP, transportando cada uno de ellos un segmento UDP diferente. El primer datagrama tiene una dirección IP de origen A1, una dirección IP de destino B, un puerto de origen P1 y un puerto de destino T. El segundo datagrama tiene la dirección IP de origen A2, la dirección IP de destino B, el puerto de origen P2 y el puerto de destino T. Suponga que A1 es diferente de A2 y que P1 es diferente de P2. Suponiendo que ambos datagramas llegan a

su destino final, ¿serán recibidos los dos datagramas UDP por el mismo socket? ¿Por qué?

b. Suponga que Alicia, Benito y Clara desean mantener una conferencia de voz utilizando SIP y RTP. Para que Alicia pueda enviar a y recibir paquetes RTP procedentes de Benito y Clara, ¿basta con un socket UDP (además del socket necesario para los mensajes SIP)? En caso afirmativo, ¿cómo distingue el cliente SIP de Alicia los paquetes RTP recibidos de Benito de los procedentes de Clara?

P17. Considere una sesión RTP que consta de cuatro usuarios, en la que todos envían y reciben paquetes RTP en la misma dirección de multidifusión. Cada usuario envía vídeo a 100 kbps.

a. ¿Limitará RTCP su tráfico a esa velocidad?
b. ¿Cuánto ancho de banda RTCP se le asignará a un receptor particular?
c. ¿Cuánto ancho de banda RTCP se le asignará a un emisor particular?

P18. a. ¿En qué se parece RTSP a HTTP? ¿RTSP tiene métodos? ¿Puede utilizarse HTTP para solicitar un flujo?

b. ¿En qué se diferencia RTSP de HTTP? Por ejemplo, HTTP es un protocolo en banda o fuera de banda? ¿Mantiene RTSP información de estado acerca del cliente (considere la función pausar/reanudar)?

P19. Verdadero o falso:

a. Si un flujo de vídeo almacenado se descarga directamente de un servidor web a un reproductor multimedia, entonces la aplicación está usando TCP como protocolo de transporte subyacente.
b. Cuando se está usando RTP, un emisor puede cambiar la codificación en mitad de una sesión.
c. Todas las aplicaciones que usan RTP deben emplear el puerto 87.
d. Si una sesión RTP tiene un flujo separado de audio y de vídeo para cada emisor, entonces los flujos de audio y de vídeo usan el mismo SSRC.
e. En los servicios diferenciados, aunque el comportamiento por salto define diferencias en el rendimiento entre clases, no se impone ningún mecanismo concreto para conseguir esos rendimientos.
f. Suponga que Alicia desea establecer una sesión SIP con Benito. Alicia, en su mensaje INVITE incluye la línea: m=audio 48753 RTP/AVP 3 (AVP 3 indica audio GSM). Alicia ha indicado, por tanto, en este mensaje que desea enviar audio GSM.
g. Respecto a la afirmación anterior, Alicia ha indicado en su mensaje INVITE que enviará el audio al puerto 48753.
h. Normalmente los mensajes SIP se envían entre entidades SIP utilizando un número de puerto SIP por defecto.
i. Para mantener el registro, los clientes SIP tienen que enviar periódicamente mensajes REGISTER.
j. SIP obliga a que todos los clientes SIP soporten la codificación de audio G.711.

P20. Suponga que se aplica la disciplina de planificación WFQ a un buffer que soporta tres clases y suponga que los pesos para las tres clases son 0,5, 0,25 y 0,25.

a. Suponga que cada clase tiene un gran número de paquetes en el buffer. ¿En qué secuencia se dará servicio a las tres clases para obtener los pesos WFQ? (En la planificación por turno rotario, una secuencia natural es 123123123 . . .)

b. Suponga que las clases 1 y 2 tienen una gran cantidad de paquetes en el buffer y que no hay paquetes de clase 3 en el buffer. ¿En qué secuencia se dará servicio a las tres clases para obtener los pesos WFQ?

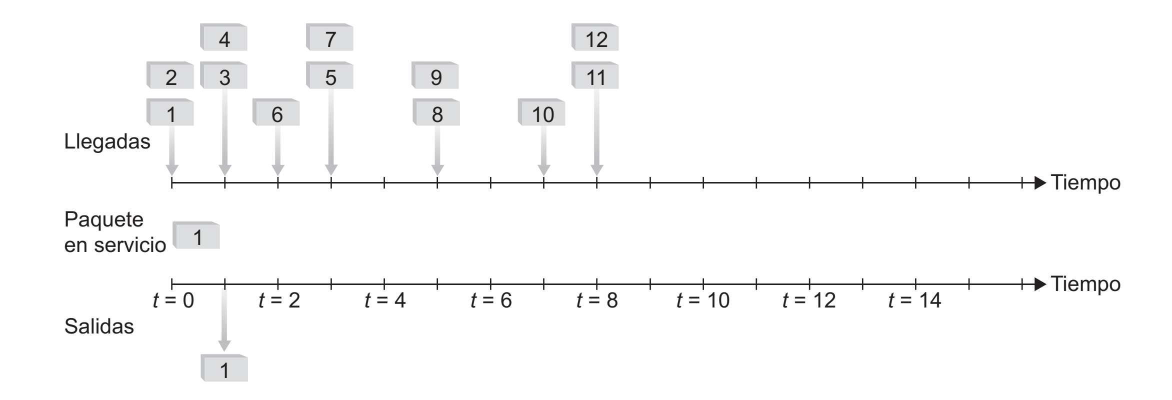
P21. Considere la figura de más abajo, que es similar a las Figuras 7.22– 7.25. Responda a las siguientes cuestiones:

a. Suponiendo un servicio FIFO, indique el instante en el que los paquetes 2 hasta 12 abandonan la cola. Para cada paquete, ¿cuál es el retardo entre su llegada y el inicio de la partición en la que se transmite? ¿Cuál es el promedio de este retardo para los 12 paquetes?

b. Suponga ahora un servicio con prioridad, en el que los paquetes con número impar tienen prioridad alta y los paquetes con número par tienen prioridad baja. Indique el instante en que los paquetes 2 a 12 abandonan la cola. Para cada paquete, ¿cuál es el retardo entre su llegada y el inicio de la partición en la que se transmiten? ¿Cuál es el promedio de este retardo para los 12 paquetes?

c. Suponga ahora un servicio por turno rotatorio. Suponga que los paquetes 1, 2, 3, 6, 11 y 12 pertenecen a la clase 1 y que los paquetes 4, 5, 7, 8, 9 y 10 son de clase 2. Indique el instante en el que cada uno de los paquetes 2 a 12 abandonan la cola. Para cada paquete, ¿cuál es el retardo entre su llegada y su salida? ¿Cuál es el promedio de este retardo para los 12 paquetes?

d. Ahora suponga un servicio WFQ (*Weighted Fair Queueing*). Suponga que los paquetes con número impar son de clase 1 y que los paquetes con número para son de clase 2. La clase 1 tiene un peso WFQ de 2 mientras que la clase 2 tiene un peso WFQ de 1. Observe que puede no ser posible conseguir una planificación WFQ idealizada como se ha descrito en el texto, por lo que debe indicar por qué ha decidido que cada paquete concreto entre en servicio en cada partición de tiempo. Para cada paquete, ¿cuál es el retardo entre su llegada y su salida? ¿Cuál es el promedio de este retardo para los 12 paquetes?



e. ¿Qué ha observado en el retardo promedio para cada uno de los cuatro casos (FCFS, por turno rotatorio, con prioridad y WFQ)?

P22. Considere de nuevo la figura del Problema P21.

a. Suponga un servicio con prioridad, teniendo los paquetes 1, 4, 5, 6 y 11 prioridad alta y los restantes paquetes prioridad baja. Indique las particiones de tiempo en las que los paquetes 2 a 12 salen de la cola.

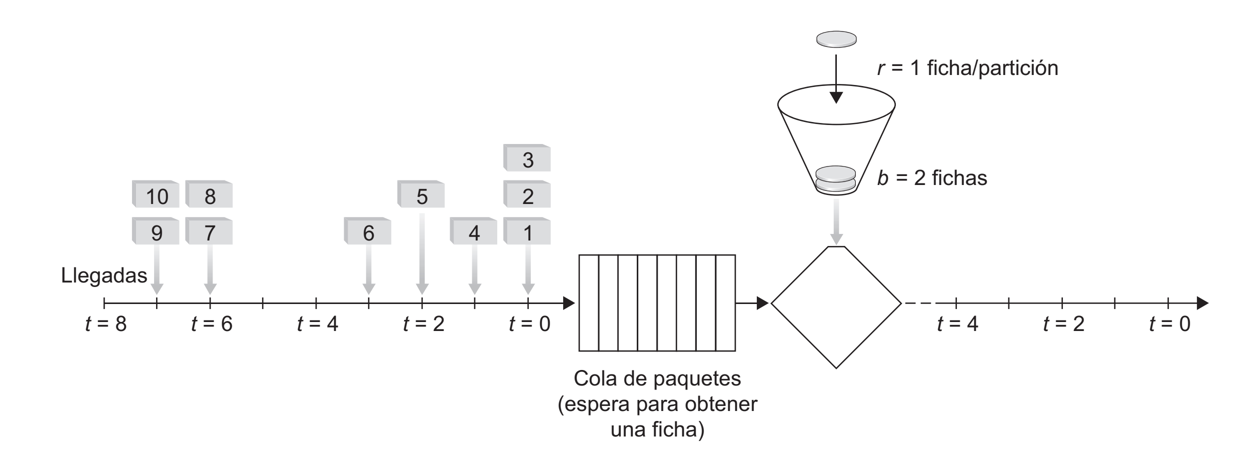
b. Suponga ahora que se utiliza un servicio por turno rotatorio, perteneciendo los paquetes 1, 4, 5, 6 y 11 a una clase de tráfico y los restantes paquetes a una segunda clase de tráfico. Indique las particiones de tiempo en las que cada uno de los paquetes 2 a 12 salen de la cola.

c. Suponga ahora que se emplea un servicio WFQ, perteneciendo los paquetes 1, 4, 5, 6 y 11 a una clase de tráfico y los restantes paquetes a una segunda clase de tráfico. La clase 1 tiene un peso WFQ de 1 y la clase 2 tiene un peso WFQ de 2 (fíjese en que estos pesos son diferentes a los de la cuestión anterior). Indique las particiones de tiempo en las que cada uno de los paquetes 2 a 12 abandonan la cola. Véase también la advertencia indicada en la cuestión anterior relativa al servicio WFQ.

P23. Considere la figura de esta página, la cual muestra una cubeta con pérdidas que está siendo alimentada con un flujo de paquetes. El buffer de fichas puede almacenar como máximo dos fichas e, inicialmente, en $t = 0$ está lleno. Las nuevas fichas llegan con una tasa igual a una ficha por partición de tiempo. La velocidad del enlace de salida es tal que si dos paquetes obtienen fichas al principio de una partición de tiempo, ambos pueden dirigirse al enlace de salida en la misma partición de tiempo. Los detalles de temporización del sistema son los siguientes:

1. Los paquetes (si los hay) llegan al principio de la partición de tiempo. Así, en la figura, los paquetes 1, 2 y 3 llegan en la partición 0. Si ya existen paquetes en la cola, entonces los paquetes que llegan se colocan al final de la cola. Los paquetes avanzan hacia el principio de la cola siguiendo una planificación FIFO.

2. Una vez que se han añadido a la cola las fichas que llegan, si existe algún paquete en la cola, uno o dos de estos paquetes (dependiendo del número de fichas disponibles) eliminarán cada uno de ellos una ficha del buffer de fichas y pasan al enlace



de salida durante dicha partición. Por tanto, los paquetes 1 y 2 eliminan cada uno una ficha del buffer (puesto que inicialmente existen dos fichas) y pasan al enlace de salida durante la partición de tiempo 0.

3. Si el buffer de fichas no está lleno se añade una nueva ficha, ya que la velocidad de generación de fichas es $r = 1$ ficha/partición.
4. El tiempo avanza a la siguiente partición de tiempo y estos pasos se repiten.

Responda a las siguientes preguntas:

a. Para cada partición de tiempo, identifique los paquetes que se encuentran en la cola y el número de fichas que hay en la cubeta inmediatamente después de que las llegadas hayan sido procesadas (paso 1 anterior) pero antes de que cualquiera de los paquetes haya atravesado la cola y eliminado una ficha. Luego, para la partición $t = 0$ del ejemplo anterior, los paquetes 1, 2 y 3 están en la cola y en el buffer hay dos fichas.

b. Para cada partición de tiempo, indique qué paquetes aparecen en la salida después de que se hayan eliminado de la cola la o las fichas. Por tanto, para la partición $t = 0$ del ejemplo anterior, los paquetes 1 y 2 aparecen en el enlace de salida del buffer con pérdidas durante la partición 0.

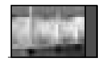
P24. Repita el Problema P23 pero suponiendo ahora que $r = 2$. Suponga de nuevo que inicialmente la cubeta está llena.

P25. Considere el Problema P24 y suponga en este caso que $r = 3$ y que $b = 2$, al igual que antes. ¿Cambiaría esto su respuesta al problema anterior?

P26. Considere el mecanismo de vigilancia de la cubeta con pérdidas (visto en la Sección 7.5) que monitoriza la velocidad promedio y el tamaño de ráfaga de un flujo de paquetes. Ahora también deseamos monitorizar la velocidad de pico, p. Explique cómo la salida de este mecanismo de vigilancia de la cubeta con pérdidas puede alimentar a un segundo mecanismo del mismo tipo, de modo que las dos cubetas con pérdidas conectadas en serie monitoricen la velocidad media, la velocidad de pico y el tamaño de ráfaga. Asegúrese de establecer el tamaño de la cubeta y la velocidad de generación de fichas del segundo mecanismo de vigilancia.

P27. Se dice que un flujo de paquetes cumple una especificación de cubeta con pérdidas (r,b) con un tamaño de ráfaga igual a b y una velocidad media r si el número de paquetes que llega a dicha cubeta es menor que $rt + b$ paquetes en todo intervalo de tiempo de longitud t para todo t. Un flujo de paquetes que cumpla con una especificación de cubeta con pérdidas (r,b), ¿tendrá que esperar alguna vez en un controlador de cubeta con pérdidas con parámetros r y b? Justifique su respuesta.

P28. Demuestre que siempre y cuando $r_1 < R\, w_1/(\sum w_j)$, entonces $d_{\text{máx}}$ es el retardo máximo que cualquier paquete del flujo 1 puede experimentar en la cola WFQ.



Preguntas para la discusión

D1. Localice una empresa que transmita flujos de vídeo en vivo utilizando mecanismos de distribución P2P. Escriba un artículo acerca de la tecnología subyacente.

D2. ¿Qué cree que es mejor, transmitir flujos de audio/vídeo almacenado sobre TCP o sobre UDP?

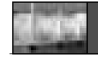
D3. Escriba un informe sobre los productos SIP de Cisco.

D4. ¿Puede resolverse el problema de proporcionar garantías de calidad de servicio (QoS) simplemente aportando el suficiente ancho de banda, es decir, actualizando las capacidades de todos los enlaces de manera que las limitaciones de ancho de banda ya no constituyan un problema?

D5. Un mercado emergente interesante es la utilización de la telefonía por Internet y una red LAN de alta velocidad por parte de una empresa para reemplazar el PBX de esa misma empresa. Escriba un informe de una página acerca de este tema. Responda a la siguientes preguntas en su informe:

a. ¿Qué es un PBX tradicional? ¿Quién lo utilizaría?
b. Considere una llamada entre un usuario de la empresa y otro usuario externo a la empresa que está conectado a la red telefónica tradicional. ¿Qué clase de tecnología se necesita en la interfaz entre la red LAN y la red telefónica tradicional?
c. Además del software de telefonía por Internet y la interfaz mencionada en el apartado (b), ¿qué más se necesita para reemplazar el PBX?

D6. Piense en cómo se dimensiona una red de carreteras (por ejemplo, cómo se determina el número de carriles de las autovías que entran y salen de una gran ciudad). Enumere los cuatro pasos que cree que seguiría un ingeniero de caminos al dimensionar tal red de carreteras. ¿Cuáles son los pasos análogos que se seguirían para dimensionar una red de computadoras?



Tareas de programación

En esta práctica de laboratorio tendrá que implementar un servidor y un cliente de flujos de vídeo. El cliente utilizará el protocolo RTSP (*Real-Time Streaming Protocol*) para controlar las acciones del servidor. El servidor utilizará el protocolo de tiempo real RTP para empaquetar el vídeo y transportarlo sobre UDP.

Se le proporcionará código Java que implementa parcialmente RTSP y RTP en el cliente y en el servidor. Su trabajo consistirá en completar tanto el código de cliente como de servidor. Cuando haya terminado, habrá creado una aplicación cliente-servidor que hará lo siguiente:

- El cliente envía comandos SETUP, PLAY, PAUSE y TEARDOWN RTSP, y el servidor responde a los comandos.
- Cuando el servidor se encuentra en el estado de reproducción, periódicamente captura una trama JPEG almacenada, la empaqueta con RTP y envía el paquete RTP a un socket UDP.
- El cliente recibe los paquetes RTP, extrae las tramas JPEG, descomprime las tramas y las reproduce en el monitor del cliente.

El código que le proporcionamos implementa el protocolo RTSP en el servidor y el desempaquetamiento RTP en el cliente. El código también se ocupa de mostrar el vídeo transmitido. Tendrá que implementar RTSP en el cliente y RTP en el servidor.

Esta tarea de programación mejorará significativamente la compresión del estudiante sobre RTP, RTSP y la transmisión de los flujos de vídeo, por lo que le recomendamos que la realice. La tarea también sugiere una serie de ejercicios opcionales, incluyendo la implementación del comando RTSP DESCRIBE tanto en el cliente como en el servidor. Puede encontrar todos los detalles acerca de la tarea, así como importantes fragmentos de código Java, en el sitio web http://www.awl.com/kurose-ross.

Henning Schulzrinne



Henning Schulzrinne es Catedrático Chair del departamento de Ciencias de la Computación y director del Internet Real-Time Laboratory en la Universidad de Columbia. Es coautor de RTP, RTSP, SIP y GIST, protocolos claves para las comunicaciones de audio y vídeo a través de Internet. Henning obtuvo su título de grado en Ingeniería Eléctrica e Industrial en la Universidad Técnica de Darmstadt, Alemania, su máster en Ingeniería Eléctrica y de computadoras en la Universidad de Cincinnati y es doctor en Ingeniería Eléctrica por la Universidad de Massachusetts, Amherst.

¿Qué le hizo especializarse en redes multimedia?

Ocurió casi por causalidad. Como estudiante de doctorado estuve incolucrado en DARTnet, una red experimental que se extendía por Estados Unidos mediante líneas T1. DARTnet se utilizó como campo de pruebas para la multidifusión y las herramientas en tiempo real de Internet. Esto me llevó a escribir mi primera herramienta para audio, NeVoT. A través de algunos de los participantes de DARTnet, me involucré en el IETF, dentro del entonces naciente grupo de trabajo Video Transport. Este grupo estandarizó más tarde RTP.

¿Cuál fue su primer trabajo en la industria de las computadoras? ¿Qué implicó?

Mi primer trabajo en la industria de las computadoras fue el de soldar un kit de computadora Altair cuando era estudiante de bachillerato en Livermore, California. Cuando volví a Alemania, comencé con una empresa pequeña de consultoría que diseñó un programa de gestión de direcciones para una agencia de viajes, almacenando los datos en cintas de casete para nuestro TRS-80 y utilizando como impresora una máquina de escribir IBM Selectric con una interfaz hardware casera.

Mi primer trabajo de verdad fue en los laboratorios AT&T Bell Laboratories, donde desarrollé un emulador de red para construir redes experimentales en un entorno de laboratorio.

¿Cuáles son los objetivos del Internet Real-Time Lab?

Nuestro objetivo es proporcionar componentes y piezas para la red Internet con el fin de que sea la única infraestructura de comunicaciones en el futuro. Esto incluye el desarrollo de nuevos protocolos como GIST (para la señalización de la capa de red) y LoST (para localizar los recursos por ubicación), o mejorar los protocolos que han estado funcionando hasta ahora, como SIP, trabajando en los sistemas Rich Presence, P2P, los mecanismos de llamadas de emergencia de siguiente generación y las herramientas de creación de servicios. Recientemente nos hemos centrado especialmente también en los sistemas inalámbricos para VoIP, como son las redes 802.11b y 802.11n y es posible también que las redes WiMax tomen importancia en las tecnologías para el campo de la telefonía. También estamos intentando mejorar de forma considerable la capacidad de los usuarios para diagnosticar los fallos en la complicada maraña de proveedores y equipos, utilizando el sistema de diagnóstico de fallos entre iguales conocido como DYSWIS (*Do You See What I See*).

Intentamos llevar a cabo el trabajo relevante de forma práctica, construyendo prototipos y sistemas de código abierto, midiendo el rendimiento de los sistemas reales y contribuyendo a los estándares del IETF.

¿Cuál es su visión del futuro de las redes multimedia?

Ahora nos encontramos en una fase de transición; estamos sólo a unos pocos años de que IP sea la plataforma universal para los servicios multimedia, desde IPTV a VoIP. Todos confiamos en que la radio, el teléfono y la televisión estén disponibles incluso durante las tormentas de nieve y los terremotos; por tanto, cuando Internet tome el papel de estas redes dedicadas, los usuarios esperarán el mismo nivel de fiabilidad.

Tendremos que aprender a diseñar tecnologías de red para un ecosistema de operadoras y proveedores de contenido y de servicios competidores, que darán servicio a muchos usuarios sin formación térchica y que tendrán que defenderse de un conjunto pequeño, pero destructivo, de usuarios maliciosos y criminales. Modificar los protocolos está comenzando a ser cada más complicado. Además, cada vez son más complejos, ya que tienen que tener en cuenta los intereses de negocios competidores, los temas de seguridad y confidencialidad y la falta de transparencia de las redes debida al uso de cortafuegos y traductores de direcciones de red.

Dado que las redes multimedia se están convirtiendo en la base de casi todas las aplicaciones de entretenimiento de gran consumo, tendrá que hacerse hincapié en la gestión de redes de muy gran tamaño, con un coste bajo. Los usuarios esperarán disponer de una gran facilidad de uso, para poder localizar, por ejemplo, el mismo contenido en todos sus dispositivos.

¿Por qué tiene SIP un futuro prometedor?

A medida que se actualicen las redes inalámbricas actuales convirtiéndose en redes 3G, es de esperar que un único mecanismo de señalización multimedia se extienda a todos los tipos de redes, desde los modems por cable a las redes telefónicas corporativas y las redes inalámbricas públicas. Junto con las radios software, esto hará posible que en el futuro pueda utilizarse un único dispositivo en una red doméstica, como por ejemplo un teléfono inalámbrico BlueTooth, en una red corporativa a través de 802.11 y en un área global mediante redes 3G. Incluso antes de que dispongamos de tal dispositivo inalámbrico universal y único, los mecanismos de movilidad personal harán posible ocultar las diferencias entre redes. Un identificador se convertirá en el medio universal de localización de una persona, en lugar de tener que recordar o transmitir media docena de números de telefóno específicos de cada tecnología o de cada ubicación.

SIP también rompe la provisión de transporte de voz (bits) desde los servicios de voz. Ahora es técnicamente posible romper con el monopolio de la telefonía local, en el que una empresa proporciona transporte de bits neutral, mientras que otros proporcionan "tono de marcado" IP y los servicios telefónicos clásicos, como las pasarelas, el reenvío de llamadas y el identificador del llamante.

Más allá de la señalización multimedia, SIP ofrece un nuevo servicio que no existía en Internet: la notificación de sucesos. Se han hecho aproximaciones a tales servicios con la técnicas HTTP y el correo electrónico, pero nunca han sido muy satisfactorias. Dado que los sucesos son una abstracción común en los sistemas distribuidos, esto puede simplificar la construcción de nuevos servicios.

¿Tiene algún consejo para los estudiantes que ahora se inician en el campo de las redes?

El campo de las redes abarca múltiples disciplinas. Deriva de la ingeniería eléctrica, las ciencias de la computación, la investigación operativa, la estadística, la economía y otras disciplinas. Por tanto, los investigadores de redes tienen que estar familizarizados con temas que van bastante más allá de los protocolos y los algoritmos de enrutamiento.

Dado que las redes se están convirtiendo en una parte importante de la vida cotidiana, los estudiantes que deseen sobresalir en este campo deben pensar en las nuevas restricciones de recursos que afectan a las redes, el tiempo y el esfuerzo de las personas, en lugar de pensar sólo en el ancho de banda o en la capacidad de almacenamiento.

Trabajar en el campo de investigación de las redes puede resultar tremendamente satisfactorio, ya que es algo que permite a las personas comunicarse e intercambiar ideas, una de las esencias de los seres humanos. Internet se ha convertido en la tercera infraestructura global más importante, junto con los sistemas de transporte y la distribución de energía. Prácticamente ninguna parte de la economía puede funcionar sin redes de altas prestaciones, por lo que el futuro próximo está repleto de oportunidades.



CAPÍTULO 8

Seguridad en las redes de computadoras

En la Sección 1.6 hemos descrito algunos de los ataques más dañinos y predominantes en Internet, incluyendo los ataques de software malicioso, de denegación de servicio, de husmeadores, de enmascaramiento de orígenes y borrado y modificación de mensajes. Aunque ya hemos estudiado un montón de cosas acerca de las redes de computadoras, no hemos examinado todavía cómo dotar de seguridad a las redes para defenderse de los ataques esbozados en la Sección 1.6. Equipados con nuestros conocimientos recién adquiridos sobre las redes de computadoras y los protocolos de Internet, ahora vamos a estudiar en profundidad las comunicaciones seguras y, en concreto, cómo las redes de computadoras pueden defenderse de los malos.

Presentemos a Alicia y Benito, dos personas que desean comunicarse y desean hacerlo "de forma segura". Puesto que éste es un texto sobre redes, debemos resaltar que Alicia y Benito podrían ser dos routers que desean intercambiar sus tablas de enrutamiento de forma segura, o un cliente y un servidor que desean establecer una conexión de transporte segura o dos aplicaciones de correo electrónico que quieren intercambiar mensajes de correo seguros (casos que consideraremos en este capítulo). Alicia y Benito son entidades bien conocidas en la comunidad de la seguridad, quizá porque sus nombres son más divertidos que una entidad genérica "A" que desea comunicarse de forma segura con una entidad genérica "B". Los asuntos amorosos ilícitos, las comunicaciones en tiempo de guerra y las transacciones de negocios son las necesidades humanas de comunicación segura habituales; prefiriendo la primera de estas necesidades a las dos últimas, nos hace felices utilizar a Alicia y Benito como nuestro emisor y nuestro receptor e imaginarlos en este primer escenario.

Hemos dicho que Alicia y Benito desean comunicarse "de forma segura", pero, ¿qué quiere decir esto exactamente? Como veremos, la seguridad (como el amor) es algo que tiene muchos matices; es decir, la seguridad tiene muchas caras. Realmente, Alicia y Benito

desean que el contenido de sus comunicaciones sea secreto y que nadie pueda conocerlo (sobre todo un esposo celoso). Probablemente, también desearán estar seguros de que cuando ellos están manteniendo una comunicación, realmente estén comunicándose el uno con el otro y que si la comunicación es alterada por un tercero, esa alteración pueda ser detectada. En la primera parte de este capítulo vamos a abordar las técnicas fundamentales de criptografía que permiten cifrar las comunicaciones, autenticar al interlocutor con el que se establece la comunicación y garantizar la integridad del mensaje.

En la segunda parte del capítulo examinaremos cómo pueden utilizarse los principios fundamentales de la criptografía para crear protocolos de red seguros. De nuevo, siguiendo el método de arriba-abajo, examinaremos los protocolos seguros en cada una de las cuatro capas superiores, comenzando por la capa de aplicación. Veremos cómo dotar de seguridad al correo electrónico, cómo dotar de seguridad a una conexión TCP, cómo proporcionar seguridad a la capa de red y cómo dotar de seguridad a una red LAN inalámbrica. En la tercera parte del capítulo consideraremos la seguridad operacional, la cual se ocupa de la protección de las redes institucionales frente a los ataques. En particular, estudiaremos cómo los cortafuegos y los sistemas de detección de intrusiones pueden mejorar la seguridad de la red de una organización.

8.1 ¿Qué es la seguridad de red?

Comencemos nuestro estudio sobre la seguridad de las redes volviendo a nuestros amantes Alicia y Benito que desean comunicarse "de forma segura". ¿Qué significa esto exactamente? En realidad, Alicia quiere que sólo Benito sea capaz de comprender los mensajes que ella le envía, incluso aunque estén comunicándose a través de un medio no seguro en el que un intruso (Tomás, por ejemplo) pueda interceptar lo que Alicia transmite a Benito. Benito también quiere estar seguro de que el mensaje que él recibe de Alicia fue realmente enviado por Alicia, y Alicia quiere estar segura de que la persona que se está comunicando con ella es realmente Benito. Alicia y Benito también quieren estar seguros de que el contenido de sus mensajes no ha sido alterado en el camino. Además, quieren estar seguros de que siempre podrán comunicarse (es decir, que nadie les puede denegar el acceso a los recursos necesarios para comunicarse). Teniendo en cuenta estas consideraciones, podemos identificar las siguientes propiedades deseables en una **comunicación segura**.

- *Confidencialidad.* Sólo el emisor y el receptor deseado deberán comprender el contenido de los mensajes transmitidos. Puesto que los curiosos pueden interceptar los mensajes, es absolutamente necesario que los mensajes sean **cifrados** de alguna manera, de modo que un mensaje interceptado no pueda ser comprendido por el que lo ha interceptado. Este aspecto de la confidencialidad es probablemente el concepto más comúnmente percibido del término *comunicación segura.* En la Sección 8.2 estudiaremos las técnicas criptográficas para el cifrado y descifrado de datos.
- *Autenticación del punto terminal.* Tanto el emisor como el receptor deberán poder confirmar la identidad del otro en el proceso de comunicación (confirmar que el otro es de hecho quien dice ser). La comunicación humana frente a frente resuelve este problema fácilmente gracias al reconocimiento visual. Cuando las entidades se comunican a través de un medio en el que no es posible ver al otro, la autenticación no es tan sencilla. Por ejemplo, ¿por qué debería creerse que un mensaje de correo electró-

nico recibido que contiene una cadena de texto que dice que dicho mensaje procede de un amigo suyo realmente procede de ese amigo?

- *Integridad del mensaje.* Incluso si el emisor y el receptor son capaces de autenticarse entre sí, continuarán queriendo estar seguros de que el contenido de sus comunicaciones no ha sido alterado durante la transmisión ni maliciosamente ni por accidente. Se pueden emplear extensiones a las técnicas de suma de comprobación que hemos visto en los protocolos de enlace de datos y de transporte fiable para proporcionar integridad a los mensajes. Estudiaremos la autenticación del punto terminal y la integridad de los mensajes en la Sección 8.3.
- *Seguridad operacional.* Casi todas las organizaciones (empresas, universidades, etc.) actuales disponen de redes que están conectadas a la red pública Internet. Estas redes pueden, potencialmente, verse comprometidas por los atacantes que pueden acceder a ellas a través de Internet. Los atacantes pueden intentar depositar gusanos en los hosts de la red, conseguir secretos corporativos, realizar un mapa de las configuraciones internas de la red y ejecutar ataques DoS. En la Sección 8.8 veremos que se emplean dispositivos operacionales, como los cortafuegos y los sistemas de detección de intrusiones, para responder a los ataques efectuados contra la red de una organización. Un cortafuegos se coloca entre la red de la organización y la red pública, controlando el acceso de paquetes procedentes de la red. Un sistema de detección de intrusiones realiza una "inspección profunda de los paquetes", alertando a los administradores de la red cuando detecta cualquier actividad sospechosa.

Una vez establecido lo que queremos decir al hablar de la seguridad de la red, vamos a ver exactamente a qué información puede tener acceso un intruso y qué acciones pueden llevar a cabo dicho intruso. La Figura 8.1 ilustra este escenario. Alicia es el emisor y desea enviar datos a Benito, que es el receptor. Para intercambiar datos de forma segura y poder cumplir los requisitos de confidencialidad, autenticación del punto terminal e integridad de los mensajes, Alicia y Benito intercambiarán mensajes de control y mensajes de datos (de forma similar a como los emisores y receptores TCP intercambian segmentos de control y segmentos de datos). Normalmente, todos o algunos de estos mensajes serán cifrados. Como hemos visto en la Sección 1.6, potencialmente un intruso puede:

- *curiosear* (husmear y registrar los mensajes de control y de datos que se transmiten por el canal).
- *modificar, insertar* o *borrar* mensajes o el contenido de los mismos.

Como veremos, a menos que se tomen las contramedidas adecuadas, estas capacidades permitirán a un intruso montar una amplia variedad de ataques contra la seguridad de la red: escuchando las comunicaciones (posiblemente robando las contraseñas y los datos), suplantando a otra entidad, pirateando una sesión activa, denegando el servicio a los usuarios legítimos de la red por sobrecarga de los recursos del sistema, etc. Puede ver un resumen de los ataques conocidos en el Centro de coordinación CERT [CERT 2009]. Consulte también [Cisco Security 2009; Voydock 1983; Bhimani 1996; Skoudis 2006].

Una vez que ha quedado claro el hecho de que existen amenazas reales en Internet, ¿cuáles son los equivalentes en Internet de Alicia y Benito, nuestros amigos que necesitan comunicarse de forma segura? Realmente, Benito y Alicia pueden ser personas situadas en dos sistemas terminales, por ejemplo, una Alicia real y un Benito real que desean intercambiar mensajes de correo electrónico de forma segura. Asimismo, también pueden desear par-

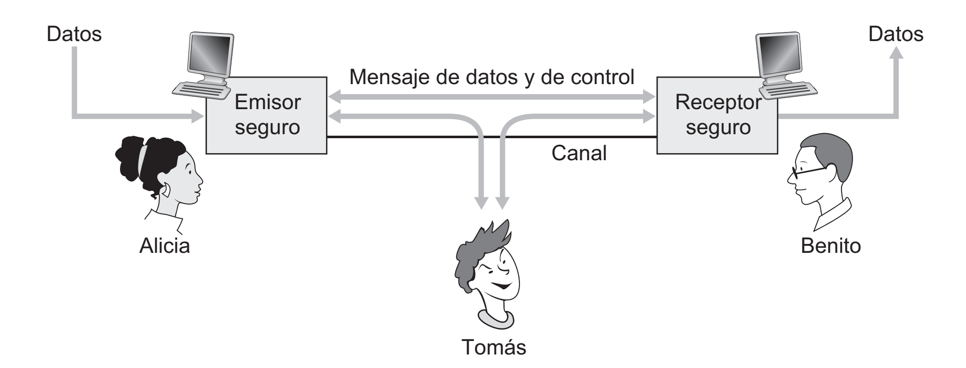


Figura 8.1 • Emisor, receptor e intruso (Alicia, Benito y Tomás).

ticipar en una transacción de comercio electrónico; por ejemplo, Benito puede tener que transferir el número de su tarjeta de crédito de forma segura a un servidor web para comprar cierto producto en línea. De forma similar, una Alicia real puede querer interactuar con su banco en línea. Los participantes que necesitan comunicaciones seguras pueden ser ellos mismos parte de la infraestructura de red. Recuerde que el sistema de nombres de dominio (DNS, véase la Sección 2.5) o los demonios de enrutamiento que intercambian información de enrutamiento (véase la Sección 4.6) requieren una comunicación segura entre ambas partes. Esto también es así en el caso de las aplicaciones de gestión de red, un tema que examinaremos en el Capítulo 9. Un intruso que pudiera interferir de forma activa en las búsquedas DNS (como hemos visto en la Sección 2.5), los cálculos de enrutamiento [Murphy 2003] o las funciones de gestión de red [RFC 2574] podría causar estragos en Internet.

Una vez que hemos establecido el marco de trabajo, algunas de las definiciones más importantes y la necesidad de dotar de seguridad a la red, vamos a profundizar a continuación en la criptografía. Aunque el uso de la criptografía para proporcionar confidencialidad es evidente, veremos también que la criptografía resulta fundamental para la autenticación del punto terminal y la verificación de la integridad de los mensajes, haciendo de ella una piedra angular de los mecanismos de seguridad de la red.

8.2 Principios de la criptografía

Aunque la criptografía tiene una larga historia que se remonta hasta la época de Julio César, las técnicas criptográficas modernas, incluyendo muchas de las utilizadas en Internet, están basadas en los avances realizados en los últimos 30 años. El libro de Kahn, *The Codebreakers* [Kahn 1967], y el libro de Singh, *The Code Book: The Science of Secrecy from Ancient Egypt to Quantum Cryptography* [Singh 1999], proporcionan una fascinante visión de la larga historia de la criptografía. Una exposición completa sobre critpografía requiere un libro completo [Kaufman 1995; Schneier 1995] y, por tanto, aquí sólo vamos a abordar los aspectos esenciales, en particular aquellos que están en práctica en Internet. Debemos comentar también que aunque en esta sección vamos a centrarnos en el uso de la criptografía para conseguir confidencialidad, veremos después que las técnicas criptográficas están inextricablemente unidas a la autenticación, la integridad de los mensajes, el no repudio y otras muchas cuestiones.

Las técnicas criptográficas permiten a un emisor ocultar los datos de modo que los intrusos no puedan obtener ninguna información a partir de los datos interceptados. El receptor, por supuesto, deberá ser capaz de recuperar los datos originales a partir de los datos ocultados. La Figura 8.2 ilustra parte de la terminología más importante.

Suponga ahora que Alicia quiere enviar un mensaje a Benito. El mensaje de Alicia en su forma original (por ejemplo, "`Benito, te quiero. Alicia`") se conoce con el nombre de **texto en claro** o **texto plano** (*cleartext* o *plaintext*) Alicia cifra su mensaje de texto en claro utilizando un **algoritmo de cifrado** de modo que el mensaje cifrado, que se conoce con el nombre de **texto cifrado** (*ciphertext*), es ininteligible para cualquier intruso. Es interesante observar que, en muchos sistemas criptográficos modernos, incluyendo los utilizados en Internet, la propia técnica de cifrado es *conocida*, en el sentido de que es pública, está estandarizada y está disponible para todo el mundo (por ejemplo, [RFC 1321; RFC 2437; RFC 2420; NIST 2001]), incluso para los potenciales intrusos). Evidentemente, si todo el mundo conoce el método utilizado para codificar los datos, entonces deberá existir algún tipo de información secreta que impida a un intruso descifrar los datos transmitidos: aquí es donde entran en acción las claves.

En la Figura 8.2 Alicia proporciona una **clave**, K_A, una cadena de números o caracteres como entrada para el algoritmo de cifrado. El algoritmo de cifrado toma la clave y el mensaje de texto en claro, *m,* como entrada y genera el texto cifrado como salida. La notación $K_A(m)$ hace referencia al formato en texto cifrado (cifrado utilizando la clave K_A) correspondiente al mensaje de texto en claro, *m*. El algoritmo de cifrado real con el que se vaya a utilizar la clave K_A resultará evidente dentro del contexto. De forma similar, Benito proporcionará una clave, K_B, al **algoritmo de descifrado**, que toma el texto cifrado y la clave de Benito como entrada y genera como salida el texto en claro original. En otras palabras, si Benito recibe un mensaje cifrado $K_A(m)$, lo descifra realizando el cálculo $K_B(K_A(m)) = m$. En los **sistemas de clave simétrica**, las claves de Alicia y de Benito son idénticas y deben mantenerse en secreto. En los **sistemas de clave pública**, se emplea una pareja de claves. Una de las claves es conocida tanto por Benito como por Alicia (de hecho es conocida por todo el mundo). La otra clave sólo es conocida por Benito o por Alicia, pero no por ambos. En las siguiente subsecciones vamos a estudiar los sistemas de clave simétrica y de clave pública más detalladamente.

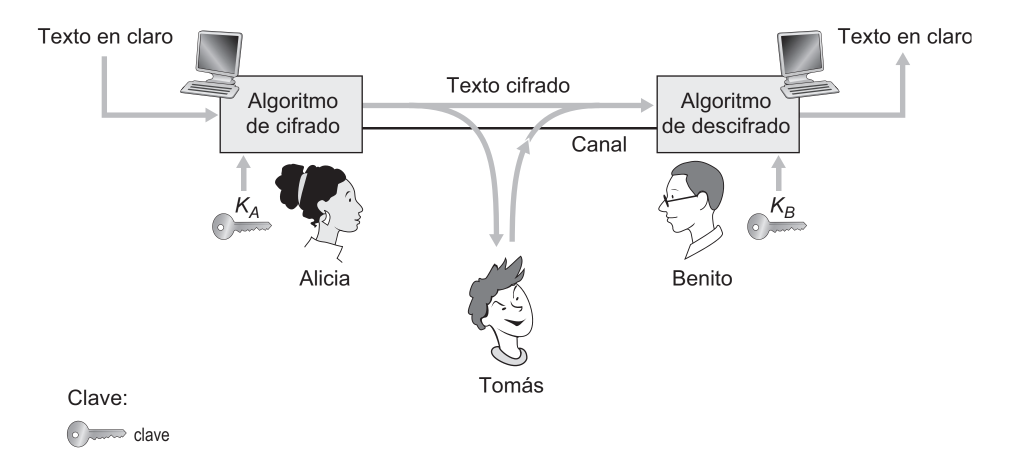


Figura 8.2 • Componentes criptográficos.

8.2.1 Criptografía de clave simétrica

Todos los algoritmos criptográficos implican sustituir una cosa por otra; por ejemplo, se toma un fragmento de texto en claro y luego se calcula y sustituye por el texto cifrado apropiado para crear el mensaje cifrado. Antes de estudiar un sistema criptográfico moderno basado en claves, entremos primero en materia estudiando un algoritmo de clave simétrica muy simple y muy antiguo atribuido a Julio César, y que se conoce con el nombre de **cifrado de César** (la palabra cifrado se emplea a menudo para designar a un método que permite encriptar los datos).

Para un texto en español, el cifrado de César funcionaría tomando cada letra del mensaje en claro y sustituyéndola por la letra que está *k* posiciones por detrás en el alfabeto (volviendo al principio una vez que se llega al final; es decir, haciendo que a la letra *z* la siga la letra *a*). Por ejemplo, si $k = 3$, entonces la letra *a* del texto en claro se convertirá en la letra *d* en el texto cifrado; la letra *b* de un texto en claro se convertirá en la letra *e* en el texto cifrado, y así sucesivamente. Aquí, el propio valor de *k* sirve como clave. Por ejemplo, el mensaje de texto en claro "`benito, te quiero. alicia`" se transformaría en "`ehqlwr, wh txlhur. dolfld`" en el texto cifrado. Aunque el texto cifrado parece una sucesión de letras sin sentido, en realidad no se tardaría mucho en romper el código si se sabe que se está utilizando el cifrado de César, ya que sólo hay 25 posibles valores de clave.

Una mejora del cifrado de César sería el **cifrado monoalfabético**, que también sustituye una letra del alfabeto por otra. Sin embargo, en lugar de efectuar esas sustituciones según una patrón regular (por ejemplo, utilizando un desplazamiento *k* igual para todas las letras), cualquier letra puede sustituirse por cualquier otra, siempre que cada una tenga una única letra y viceversa. La regla de sustitución de la Figura 8.3 muestra una posible codificación para el texto en claro.

El mensaje de texto en claro "`benito, te quiero. alicia`" se convierte en "`ncjsuk, uc pyscok. mgsbsm`". De nuevo, como en el caso del cifrado de César, esto parece una serie de letras sin sentido. El cifrado monoalfabético es desde luego mejor que el cifrado de César, en el sentido de que existen 26! (del orden de 10^{26}) posibles parejas de letras en lugar de las 25 posibles parejas que el cifrado de César proporciona. Un ataque por fuerza bruta que consistiera en probar todas las 10^{26} posibles parejas requeriría demasiado trabajo como para considerarlo una forma factible de romper el algoritmo de cifrado y decodificar el mensaje. Sin embargo, el análisis estadístico del idioma utilizado en el texto en claro, por ejemplo, saber que las letras *e* y *a* son las letras que más frecuentemente aparecen en un texto típico en español (representando aproximadamente el 14 y el 12 por ciento, respectivamente), y saber que existen determinadas combinaciones de dos y tres letras que aparecen muy frecuentemente juntas (por ejemplo, "se", "es" "re", "la" "al", etc.) puede romper de forma relativamente fácil este código. Si el intruso tiene un conocimiento acerca del posible contenido del mensaje, entonces es todavía más fácil romper el código. Por ejemplo, si el intruso Tomás es el esposo de Alicia y sospecha que ésta tiene una relación sentimental con Benito, entonces podría deducir que los nombres "benito" y "alicia" aparecerán en el texto.

```
Letras texto en claro:  a b c d e f g h i j k l m n o p q r s t u v w x y z
Letras texto cifrado:   m n b v c x z a s d f g h j k l p o i u y t r e w q
```

Figura 8.3 • Cifrado monoalfabético.

Si Tomás estuviera seguro de que esos dos nombres aparecen en el texto cifrado y dispusiera de una copia del mensaje de texto cifrado de ejemplo que hemos proporcionado anteriormente, entonces podría determinar inmediatamente doce de las 26 parejas de letras, lo que implica reducir según un factor de 10^{15} menos posibilidades que habría que comprobar mediante un método de fuerza bruta. De hecho, si Tomás sospechara que Alicia le está engañando, también esperaría encontrar algunas otras palabras seleccionadas dentro del mensaje.

A la hora de considerar lo fácil que puede ser que Tomás rompa el esquema de cifrado utilizado por Benito y Alicia, podemos distinguir tres escenarios distintos dependiendo de la información de la que disponga el intruso.

- *Ataque de sólo texto cifrado.* En algunos casos, el intruso puede tener acceso únicamente al texto cifrado interceptado, sin disponer de información segura acerca del contenido del mensaje en claro. Ya hemos visto cómo el análisis estadístico puede ayudarnos a realizar un **ataque de sólo texto cifrado** al esquema de cifrado.
- *Ataque de texto en claro conocido.* Anteriormente hemos visto que si Tomás estuviera seguro, de alguna manera, de que las palabras "benito" y "alicia" aparecen en el mensaje de texto cifrado, entonces podría haber determinado las parejas (texto en claro, texto cifrado) para las letras *a, l, i, c, b, e, n, t* y *o*. Tomás también podría haber sido lo suficientemente afortunado como para haber grabado todas las transmisiones de texto cifrado y luego encontrar en una hoja de papel la propia versión descifrada por Alicia de una de las transmisiones cifradas. Cuando un intruso conoce alguna de las parejas (texto en claro, texto cifrado), decimos que se trata de un **ataque de texto en claro conocido** al esquema de cifrado.
- *Ataque de texto en claro seleccionado.* En un **ataque de texto en claro seleccionado**, el intruso tiene la posibilidad de elegir el mensaje de texto en claro y obtener su correspondiente texto cifrado. Para los algoritmos de cifrado simples que hemos visto ahora, si Tomás pudiera hacer que Alicia enviara el mensaje "`Es extraño mojar queso en la cerveza o probar whisky de garrafa`" podría romper completamente el esquema de cifrado. Pronto veremos que, para técnicas de cifrado más sofisticadas, un ataque de texto en claro seleccionado no implica necesariamente que la técnica de cifrado pueda ser rota.

Hace quinientos años se invento una técnica denominada **cifrado polialfabético**, que permitía mejorar el cifrado monoalfabético. La idea subyacente al cifrado polialfabético es utilizar varios cifrados monoalfabéticos, utilizando un cifrado monoalfabético específico para codificar cada letra situada en una posición específica dentro del mensaje de texto en claro. De ese modo, una misma letra que aparezca en diferentes posiciones dentro del mensaje en claro se podría codificar de forma distinta cada vez. En la Figura 8.4 se muestra un ejemplo de un esquema de cifrado polialfabético. Está compuesto por dos cifrados de César (con $k = 5$ y $k = 19$), mostrados en sendas filas. Podríamos decidir utilizar estos dos cifrados de César, C_1 y C_2, según el patrón repetitivo C_1, C_1, C_2, C_1, C_2. De este modo, la primera y la segunda letras del texto en claro se codificarían utilizando C_1, la tercera aplicando C_2, la cuarta utilizando C_1 y la quinta utilizando C_2. A continuación el patrón se repite, lo que haría que la sexta letra se codificara utilizando C_1, la séptima con C_1, y así sucesivamente. Así, el mensaje de texto en claro "`benito, te quiero.`" tendría el equivalente de texto cifrado "`gjgnmt, yx vnnjkt.`" Observe que la primera *e* del mensaje de texto en claro se cifra utilizando C_1, mientras que la segunda *e* se cifra utilizando C_2. En este ejemplo, la "clave"

Letras texto en claro:	a b c d e f g h i j k l m n o p q r s t u v w x y z
$C_1(k = 5)$:	f g h i j k l m n o p q r s t u v w x y z a b c d e
$C_2(k = 19)$:	t u v w x y z a b c d e f g h i j k l m n o p q r s

Figura 8.4 • Cifrado polialfabético utilizando dos cifrados de César.

de cifrado y de descifrado es el propio conocimiento de los dos cifrados de César ($k = 5$, $k = 19$) y del patrón C_1, C_1, C_2, C_1, C_2.

Cifrado de bloque

Avancemos ahora hacia tiempos más modernos y veamos cómo se lleva a cabo hoy día el cifrado de clave simétrica. Existen dos clases generales de técnicas de cifrado simétrico: **cifrados de flujo** y **cifrados de bloque**. Examinaremos brevemente los cifrados de flujo en la Sección 8.7 cuando investiguemos la seguridad en las redes LAN inalámbricas. En esta sección, nos centraremos en los cifrados de bloque que se emplean en muchos protocolos seguros de Internet, incluyendo PGP (para correo electrónico seguro), SSL (para dotar de seguridad a las conexiones TCP) e IPsec (para dotar de seguridad al transporte de la capa de red).

En un cifrado de bloque, el mensaje que hay que cifrar se procesa en bloques de k bits. Por ejemplo, si $k = 64$, entonces el mensaje se descompone en bloques de 64 bits y cada bloque se cifra de forma independiente. Para codificar un bloque, el sistema de cifrado asigna una correspondencia uno-a-uno, con el fin de asignar el bloque de k bits de texto en claro a un bloque de k bits de texto cifrado. Veamos un ejemplo. Suponga que $k = 3$, de modo que el cifrado de bloque asigna a cada entrada de 3 bits (texto en claro) una salida de 3 bits (texto cifrado). En la Tabla 8.1 se proporciona una posible de estas correspondencias. Observe que se trata de una aplicación uno-a-uno, es decir, existe una salida diferente para cada entrada. Este cifrado de bloque descompone el mensaje en bloques de 3 bits y cifra cada bloque de acuerdo con la correspondencia anterior. Puede verificar que si se cifra el mensaje 010110001111 se obtiene como resultado 101000111001.

Continuando con este ejemplo de bloque de 3 bits, observe que la correspondencia de la Tabla 8.1 es simplemente una asignación de entre las muchas posibles. ¿Cuántas posibles correspondencias existen? Para responder a esta cuestión simplemente observe que una correspondencia no es otra cosa que una permutación de todas las posibles entradas. Existen 2^3 (= 8) posibles entradas (enumeradas bajo las columnas etiquetadas como Entrada). Estas ocho entradas pueden permutarse en 8! = 40.320 formas distintas. Dado que cada una de estas permutaciones especifica una correspondencia, habrá 40.320 posibles correspondencias. Podemos interpretar cada una de estas correspondencias como una clave: si Alicia y Benito conocen simultáneamente la correspondencia (la clave), podrán cifrar y descifrar los mensajes que se intercambien.

El ataque por fuerza bruta a este sistema de cifrado consistiría en tratar de descifrar un texto utilizando todas las correspondencias posibles. Puesto que sólo existen 40.320 correspondencias (cuando $k = 3$), esto puede llevarse a cabo rápidamente con cualquier PC de escritorio. Para evitar los ataques por fuerza bruta, los sistemas de cifrado de bloque normalmente utilizan bloques de mucho mayor tamaño, compuestos de $k = 64$ bits o incluso mayores. Observe que el número de correspondencias posibles para un cifrado cualquiera

Entrada	Salida	Entrada	Salida
000	110	100	011
001	111	101	010
010	101	110	000
011	100	111	001

Tabla 8.1 • Un cifrado de bloque específico de 3 bits.

de bloques de k bits es de $2^k!$, que es un valor astronómico incluso para valores moderados de k (como por ejemplo $k = 64$).

Aunque los cifrados de bloque de tabla completa, como el que acabamos de describir, pueden producir esquemas de cifrado de clave simétrica robustos, incluso con valores moderados de k, lamentablemente resultan difíciles de implementar. Para $k = 64$ y una correspondencia determinada, Alicia y Benito necesitarían mantener una tabla con 2^{64} valores de entrada, lo que es una tarea imposible. Además, si Alicia y Benito quisieran cambiar de clave, ambos tendrían que volver a generar la tabla. Por tanto, la utilización de un cifrado de bloque de tabla completa que proporcione correspondencias predeterminadas entre todas las entradas y las salidas (como en el ejemplo anterior) resulta simplemente impracticable.

En lugar de ello, los sistemas de cifrado de bloque suelen utilizar funciones que simulan la generación de tablas aleatoriamente permutadas. En la Figura 8.5 se muestra un ejemplo (adaptado de [Kaufman 1995]) de una función de este tipo para $k = 64$. La función descompone en primer lugar el bloque de 64 bits en ocho fragmentos, estando cada fragmento compuesto por 8 bits. Cada fragmento de 8 bits se procesa mediante una tabla de 8 por 8 bits, que tiene un tamaño manejable. Por ejemplo, el primer fragmento se procesa

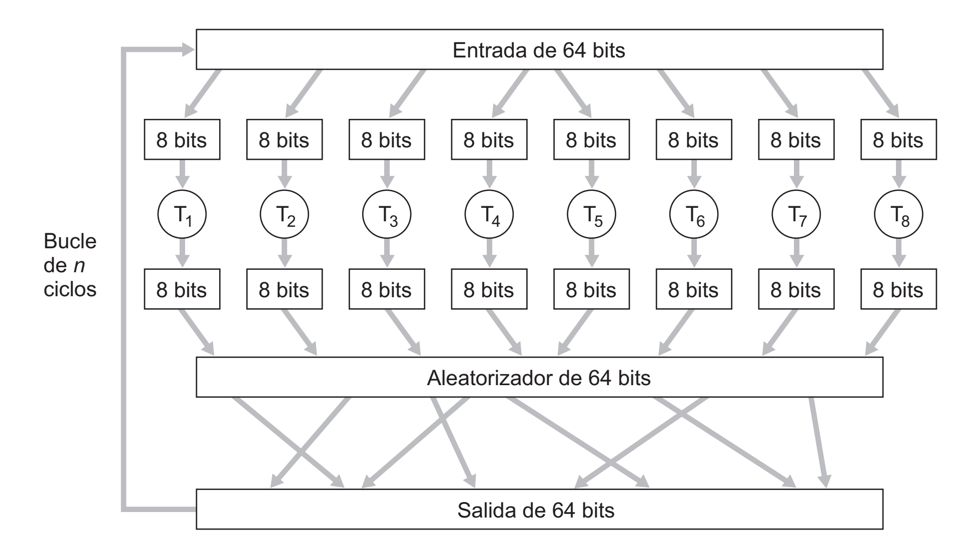


Figura 8.5 • Ejemplo de un cifrado de bloque.

mediante la tabla designada como T_1. A continuación, los 8 fragmentos de salida se recomponen en un bloque de 64 bits. Las posiciones de los 64 bits del bloque a continuación se aleatorizan (permutan) para generar una salida de 64 bits. Esta salida se realimenta hacia la entrada de 64 bits, donde comienza un nuevo ciclo. Después de *n* ciclos como éste, la función proporciona un bloque de 64 bits de texto cifrado. El propósito de los distintos ciclos es hacer que cada bit de entrada afecte a la mayoría (si no a todos) de los bits de salida finales (si sólo se utilizara un ciclo, cada bit de entrada dado sólo afectaría a 8 de los 64 bits de salida). La clave para este algoritmo de cifrado de bloque estaría compuesta por las ocho tablas de permutación (asumiendo que la función de aleatorización sea de dominio público).

Hoy día existen varios sistemas de cifrado de bloque populares, incluyendo DES (*Data Encryption Standard*, Estándar de cifrado de datos), 3DES y AES (*Advanced Encryption Standard*, Estándar avanzado de cifrado). Cada uno de estos estándares utiliza funciones en lugar de tablas predeterminadas, según las ideas expuestas en la Figura 8.5 (aunque son más complicadas y específicas de cada sistema de cifrado). Cada uno de estos algoritmos utiliza también una cadena de bits como clave. Por ejemplo, DES emplea bloques de 64 bits con una clave de 56 bits. AES utiliza bloques de 128 bits y puede operar con claves de 128, 192 y 256 bits de longitud. La clave de un algoritmo determina las "mini-tablas" específicas de asignación y permutación dentro del algoritmo. El ataque por fuerza bruta a cada uno de estos sistemas de cifrado consistiría en ir aplicando sucesivamente todas las claves, usando el algoritmo de descifrado de cada una. Observe que con una longitud de clave igual a n, existen 2^n claves posibles. NIST [NIST 2001] estima que una máquina que pudiera romper el algoritmo DES de 56 bits en un segundo (es decir, que pudiera probar en un segundo las 2^{56} claves) necesitaría aproximadamente 149 trillones de años para romper una clave AES de 128 bits.

Encadenamiento de bloques cifrados

En las aplicaciones de redes de computadoras, normalmente es necesario cifrar mensajes de gran tamaño (o largos flujos de datos). Si aplicamos un cifrado de bloques como el descrito, descomponiendo simplemente el mensaje en bloques de *k* bits y cifrando independientemente cada bloque, aparece un problema bastante sutil pero de gran importancia. Para ver en qué consiste, observe que dos o más de los bloques del texto en claro podrían ser idénticos. Por ejemplo, el texto en claro en dos o más bloques podría ser "HTTP/1.1". Para estos bloques idénticos, el cifrado de bloque produciría, por supuesto, el mismo texto cifrado. De ese modo, un atacante podría posiblemente adivinar el texto en claro cuando viera bloques de texto cifrado idénticos y podría incluso ser capaz de descifrar el mensaje completo identificando bloques de texto cifrado idénticos y utilizando el conocimiento acerca de la estructura de protocolos subyacente [Kaufman 1995].

Para resolver este problema, podemos introducir cierta aleatoriedad en el texto cifrado, de modo que idénticos bloques de texto en claro produzcan bloques de texto cifrado diferentes. Para explicar esta idea, sea $m(i)$ el *i-ésimo* bloque de texto en claro, sea $c(i)$ el *i-ésimo* bloque de texto cifrado y sea $a \oplus b$ la operación OR-exclusiva (XOR) de dos cadenas de bits, a y b. (Recuerde que $0 \oplus 0 = 1 \oplus 1 = 0$ y que $0 \oplus 1 = 1 \oplus 0 = 1$, y que la operación XOR de dos cadenas de bits se realiza bit por bit. Así, por ejemplo, $10101010 \oplus 11110000 = 01011010$.) Asimismo, designaremos mediante K_S al algoritmo de encriptación de bloque cifrado con clave S. La idea básica es la siguiente: el emisor genera un número aleatorio de k bits $r(i)$ para el *i-ésimo* bloque y calcula $c(i) = K_S(m(i) \oplus r(i))$.

Observe que se selecciona un nuevo número aleatorio de k bits para cada bloque. El emisor envía entonces $c(1)$, $r(1)$, $c(2)$, $r(2)$, $c(3)$, $r(3)$, etc. Puesto que el receptor recibe $c(i)$ y $r(i)$ puede recuperar cada bloque del texto en claro calculando $m(i) = K_S(c(i)) \oplus r(i)$. Es importante observar que, aunque $r(i)$ se envíe sin cifrar y por tanto podría ser husmeado por Tomás, éste no podrá obtener el texto en claro $m(i)$, ya que no conoce la clave K_S. Observe también que si dos bloques de texto en claro $m(i)$ y $m(j)$ son iguales, los correspondientes bloques en texto cifrado $c(i)$ y $c(j)$ serán diferentes (siempre y cuando los números aleatorios $r(i)$ y $r(j)$ sean distintos, lo que ocurre con una muy alta probabilidad).

Por ejemplo, considere el sistema de cifrado de bloques de 3 bits de la Tabla 8.1. Suponga que el texto en claro es 010010010. Si Alicia cifra esta secuencia directamente sin incluir ninguna aleatoriedad, el texto cifrado resultante será 101101101. Si Tomás captura el texto cifrado, dado que cada uno de los tres bloques de cifrado es igual, podrá suponer correctamente que cada uno de los tres bloques de texto en claro son también coincidentes. Ahora suponga que en lugar de ello Alicia genera los bloques aleatorios $r(1) = 001$, $r(2) = 111$ y $r(3) = 100$ y aplica la técnica anteriormente explicada para generar el texto cifrado $c(1) = 100$, $c(2) = 010$ y $c(3) = 000$. Observe que ahora los tres bloques de texto cifrado son distintos incluso aunque los bloques de texto en claro son iguales. Alicia envía entonces $c(1)$, $r(1)$, $c(2)$ y $r(2)$. El lector puede verificar que Benito podrá obtener el texto en claro original utilizando la clave compartida K_S.

Algún lector más astuto habrá deducido que el introducir la aleatoriedad resuelve un problema, pero crea otro: en concreto, Alicia tiene que transmitir el doble de bits que antes. De hecho, por cada bit de cifrado, Alicia debe ahora enviar también un bit aleatorio, duplicando así el ancho de banda requerido. Si lo que queremos es estar en misa y repicando a la vez, los sistemas de cifrado de bloque suelen utilizar una técnica denominada **Encadenamiento de bloques cifrados (CBC, *Cipher Block Chaining*)**. La idea básica consiste en enviar sólo *un valor aleatorio junto con el primer mensaje, y hacer que el emisor y el receptor utilicen los bloques codificados calculados, en lugar de los subsiguientes números aleatorios.* Específicamente, CBC opera como sigue:

1. Antes de cifrar el mensaje (o el flujo de datos), el emisor genera una cadena aleatoria de k bits, denominada **Vector de inicialización (IV, *Initialization Vector*)**. Denotaremos a este vector de inicialización mediante $c(0)$. El emisor envía el vector IV al receptor *sin cifrar.*
2. Para el primer bloque, el emisor calcula $m(1) \oplus c(0)$, es decir, calcula la operación OR exclusiva del primer bloque de texto en claro con IV. A continuación, introduce el resultado en el algoritmo de cifrado de bloque, para obtener el correspondiente bloque de texto cifrado; es decir, $c(1) = K_S(m(1) \oplus c(0))$. El emisor envía después el bloque cifrado $c(1)$ al receptor.
3. Para el *i-ésimo* bloque, el emisor genera el *i-ésimo* bloque de texto cifrado utilizando la fórmula $c(i) = K_S(m(i) \oplus c(i - 1))$.

Examinemos ahora algunas de las consecuencias de este método. En primer lugar, el receptor continuará pudiendo recuperar el mensaje original. De hecho, cuando el receptor reciba $c(i)$, descifrará el mensaje con K_S para obtener $s(i) = m(i) \oplus c(i - 1)$; puesto que el receptor también conoce $c(i - 1)$, puede entonces obtener el bloque de texto en claro a partir de la fórmula $m(i) = s(i) \oplus c(i - 1)$. En segundo lugar, incluso si dos bloques de texto en claro son idénticos, los textos cifrados correspondientes serán (casi siempre) diferentes. En

tercer lugar, aunque el emisor envíe el vector IV sin cifrar, ningún intruso podrá descifrar los bloques de texto cifrado dado que no conocen la clave secreta, *S*. Por último, el emisor sólo envía un bloque de sobrecarga (el vector IV), con lo que el uso de ancho de banda sólo se incrementa de una forma prácticamente despreciable, asumiendo que estemos utilizando mensajes de gran longitud (compuestos de centenares de bloques).

Como ejemplo, determinemos ahora el texto cifrado para el sistema de cifrado de bloque de 3 bits de la Tabla 8.1, utilizando como texto en claro 010010001 y como vector IV = $c(0)$ = 001. En primer lugar, el emisor utiliza el vector IV para calcular $c(1) = K_S(m(1) \oplus c(0))$ = 100 y a continuación calcula $c(2) = K_S(m(2) \oplus c(1)) = K_S(010 \oplus 100) = 000$ y $c(3) = K_S(m(3) \oplus c(2)) = K_S(010 \oplus 000) = 101$. El lector puede verificar que el receptor, conociendo el vector IV y la clave K_S, puede recuperar el texto en claro original.

La técnica CBC tiene una importante consecuencia a la hora de diseñar protocolos de red seguros: necesitaremos proporcionar un mecanismo dentro del protocolo para transferir el vector IV desde el emisor al receptor. Posteriormente en el capítulo veremos cómo se hace esto en diversos protocolos.

8.2.2 Cifrado de clave pública

Durante más de 2.000 años (desde la época del cifrado de César hasta la década de 1970), la comunicación cifrada requería que los dos interlocutores que se estaban comunicando compartieran una clave secreta: la clave simétrica utilizada para el cifrado y el descifrado. Una dificultad con esta técnica es que ambas partes deben acordar de alguna manera cuál es esa clave secreta, ¡pero el hacer eso requiere que se comuniquen (presumiblemente de forma *segura*)! Quizá ambas partes podrían primero reunirse y acordar en persona cuál es esa clave (por ejemplo, dos de los centuriones de César podrían reunirse en una termas romanas) y en lo sucesivo comunicarse mediante un método de cifrado. Sin embargo, en un mundo conectado en red, los interlocutores que se están comunicando pueden no llegar a encontrarse nunca de forma física, y puede que incluso no lleguen a conversar excepto a través de la red. ¿Es posible que dos partes se comuniquen de forma cifrada sin conocer de antemano una clave secreta compartida? En 1976 Diffie y Hellman [Diffie 1976] inventaron un algoritmo (que ahora se conoce como algoritmo de intercambio de claves de Diffie-Hellman) para hacer precisamente eso; se trata de un enfoque radicalmente distinto y maravillosamente elegante para las comunicaciones seguras y que ha conducido al desarrollo de los sistemas actuales de clave pública. Como veremos en breve, los sistemas criptográficos de clave pública disfrutan también de diversas propiedades muy atractivas que los hacen útiles no sólo para el cifrado, sino también para la autenticación y las firmas digitales. Es interesante observar que recientemente ha salido a la luz que unas ideas similares a las de [Diffie 1976] y [RSA 1978] habían sido desarrolladas independientemente a principios de la década de 1970 en una serie de informes secretos elaborados por investigadores del Grupo de seguridad de electrónica y comunicaciones en el Reino Unido [Ellis 1987]. Como a menudo suele suceder, las grandes ideas pueden surgir de forma independiente en muchos lugares distintos; afortunadamente, los avances relativos a los sistemas de clave pública no sólo tuvieron lugar en privado, sino también a la vista del público.

Conceptualmente, la utilización de un sistema de criptografía de clave pública es muy simple. Suponga que Alicia quiere comunicarse con Benito. Como se muestra en la Figura 8.6, en lugar de que Benito y Alicia compartan una única clave secreta (como es el caso en los sistemas de clave simétrica), Benito (el receptor de los mensajes de Alicia) dispone en su

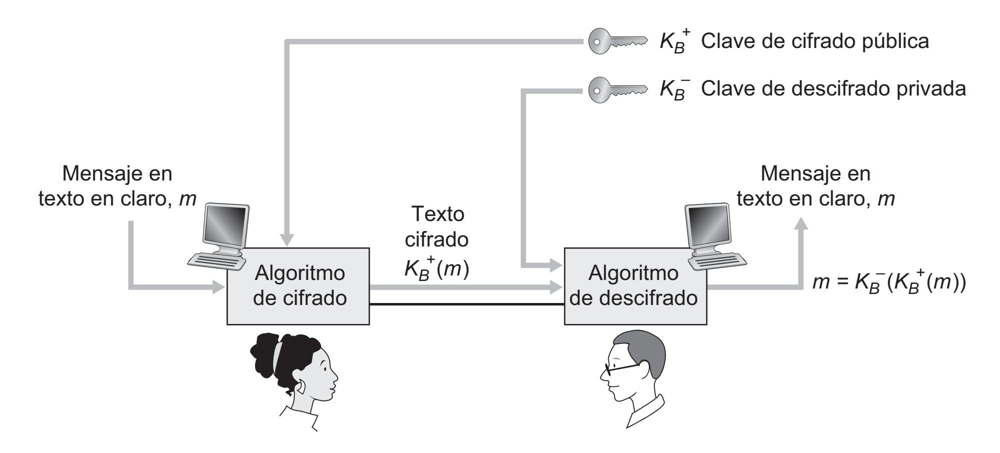


Figura 8.6 • Criptografía de clave pública.

lugar de dos claves: una **clave pública** que está disponible para *todo* el mundo (incluyendo a Tomás el intruso) y una **clave privada** que sólo Benito conoce. Utilizaremos la notación K_B^+ y K_B^- para hacer referencia a las claves pública y privada de Benito, respectivamente. Para poder comunicarse con Benito, Alicia consulta primero la clave pública de Benito y luego cifra su mensaje, *m,* destinado a Benito utilizando esa clave pública de Benito y un algoritmo de cifrado conocido (por ejemplo, un algoritmo estandarizado); es decir, Alicia calcula $K_B^+(m)$. Benito recibe el mensaje cifrado de Alicia y utiliza su clave privada y un algoritmo de descifrado conocido (por ejemplo, un algoritmo estandarizado) para descifrar el mensaje cifrado de Alicia. Es decir, Benito calcula $K_B^-(K_B^+(m))$. Como veremos más adelante, existen técnicas y algoritmos de cifrado/descifrado para seleccionar claves públicas y privadas tales que $K_B^-(K_B^+(m)) = m$; es decir, tales que al aplicar la clave pública de Benito, K_B^+, a un mensaje, *m* (para obtener $K_B^+(m)$), y aplicar luego la clave privada de Benito, K_B^-, a la versión cifrada de *m* (es decir, calcular $K_B^-(K_B^+(m))$) se vuelve a obtener *m.* Es un resultado realmente maravilloso. De esta manera, Alicia puede utilizar la clave de Benito que está públicamente disponible con el fin de enviar un mensaje secreto a Benito, sin que ninguno de los dos tenga que distribuir ninguna clave secreta. Como veremos enseguida, podemos intercambiar el papel de la clave pública y la clave privada y obtener el mismo resultado; es decir, $K_B^- (K_B{}^+(m)) = K_B^+ (K_B^-(m)) = m$.

La utilización de la criptografía de clave pública es por tanto conceptualmente muy simple. Pero puede que al lector le surjan inmediatamente dos preguntas. Un primer posible problema es que, aunque un intruso que intercepte el mensaje de cifrado de Alicia sólo obtendrá datos sin sentido, ese intruso conoce tanto la clave (la clave pública de Benito, que está disponible para que todo el mundo la vea), como el algoritmo que Alicia ha utilizado para el cifrado. Tomás podría entonces montar un ataque de texto claro conocido, utilizando ese algoritmo de cifrado estandarizado y la clave de cifrado de Benito, públicamente disponible, para codificar cualquier mensaje que desee. Tomás también podría intentar, por ejemplo, codificar mensajes, o partes de mensajes, que piense que Alicia podría enviar, con el fin de suplantarla. Obviamente, para que la criptografía de clave pública pueda funcionar, la selección de claves y el cifrado/descifrado deben hacerse de forma tal que sea imposible (o al menos lo suficientemente difícil como para ser prácticamente imposible) para un intruso determinar la clave privada de Benito o descifrar o adivinar de alguna otra manera el men-

saje que Alicia le ha enviado a Benito. El segundo problema potencial es que dado que la clave de cifrado de Benito es pública, cualquiera puede enviar un mensaje cifrado a Benito, incluyendo Alicia o alguien que *se haga pasar* por Alicia. Sin embargo, en el caso de una única clave secreta compartida, éste ya no será el caso puesto que nadie puede enviar un mensaje cifrado a Benito utilizando la clave públicamente disponible de éste. Es necesaria una firma digital, tema que estudiaremos en la Sección 8.3, para vincular a un emisor con un mensaje.

RSA

Aunque pueden existir muchos algoritmos que se correspondan con la descripción realizada, el **algoritmo RSA** (llamado así por sus inventores, Ron Rivest, Adi Shamir y Leonard Adleman) se ha convertido casi en sinónimo de la criptografía de clave pública. En primer lugar vamos a ver cómo funciona RSA y luego examinaremos por qué funciona.

RSA hace un extenso uso de las operaciones aritméticas módulo n. Por ello, vamos a repasar brevemente la aritmética modular. Recuerde que $x \bmod n$ simplemente indica el resto de dividir x entre n; así, por ejemplo, 19 mod 5 = 4. En la aritmética modular, se realizan las operaciones usuales de suma, multiplicación y exponenciación. Sin embargo, el resultado de cada operación se sustituye por el resto entero que se obtiene al dividir el resultado entre n. La realización de las operaciones de suma y multiplicación en aritmética modular se facilita teniendo en cuenta las siguientes fórmulas de utilidad:

$$[(a \bmod n) + (b \bmod n)] \bmod n = (a + b) \bmod n$$
$$[(a \bmod n) - (b \bmod n)] \bmod n = (a - b) \bmod n$$
$$[(a \bmod n) \cdot (b \bmod n)] \bmod n = (a \cdot b) \bmod n$$

Se deduce de la tercera fórmula que $(a \bmod n)^d \bmod n = a^d \bmod n$, que es una identidad que, como pronto veremos, resulta muy útil.

Supongamos ahora que Alicia desea enviar a Benito un mensaje cifrado con RSA, como se muestra en la Figura 8.6. En esta exposición sobre RSA, hay que tener siempre en mente que un mensaje no es nada más que un patrón de bits y que cada patrón de bits puede representarse de manera unívoca mediante un número entero (junto con la longitud del patrón de bits). Por ejemplo, suponga un mensaje igual al patrón de bits 1001; este mensaje puede representarse mediante el entero decimal 9. Por tanto, cifrar un mensaje con RSA es equivalente a cifrar el número entero único que representa a dicho mensaje.

En RSA existen dos componentes interrelacionados:

- La elección de las claves pública y privada.
- El algoritmo de cifrado y descifrado.

Para generar las claves RSA pública y privada, Benito lleva a cabo los pasos siguientes:

1. Elige dos números primos grandes, p y q. ¿Cómo de grandes tienen que ser p y q? Cuanto más grandes sean estos valores, más difícil será romper el algoritmo RSA, pero también se tardará más en realizar la codificación y la decodificación. RSA Laboratories recomienda que el producto de p por q sea del orden de 1.024 bits. Si desea obtener información acerca de cómo determinar números primos grandes, consulte [Caldwell 2007].

2. Calcula $n = pq$ y $z = (p - 1)(q - 1)$.
3. Elige un número, e, menor que n, que no tiene ningún factor común (distinto de 1) con z. (En este caso, se dice que e y z son números primos relativos.) Se emplea la letra e porque este valor se utilizará en el cifrado o encriptación.
4. Determina un número, d, tal que $ed - 1$ es divisible de forma exacta por z (es decir, su resto es igual a cero). La letra d se emplea porque este valor se utilizará en el descifrado. Dicho de otra forma, dado e, seleccionamos d tal que

$$ed \bmod z = 1$$

5. La clave pública que Benito pone a disposición de todo el mundo, K_B^+, es la pareja de números (n, e); su clave privada, K_B^-, es la pareja de números (n, d).

El cifrado por parte de Alicia y el descifrado que lleva a cabo Benito se realizan como sigue:

- Suponga que Alicia desea enviar a Benito un patrón de bits representado por el número entero m (con $m < n$). Para realizar la codificación, Alicia lleva a cabo la operación de exponenciación m^e, y luego calcula el resto entero que se obtiene al dividir m^e entre n. En otras palabras, el valor cifrado, c, del mensaje de texto en claro de Alicia, m, es

$$c = m^e \bmod n$$

  El patrón de bits correspondiente a este texto cifrado c se envía a Benito.

- Para descifrar el mensaje de texto cifrado recibido, c, Benito hace el cálculo siguiente:

$$m = c^d \bmod n$$

  que requiere el uso de su clave privada (n, d).

Veamos un ejemplo simple de RSA. Suponga que Benito elige $p = 5$ y $q = 7$. (Evidentemente, estos valores son demasiado pequeños como para ser seguros.) Entonces, $n = 35$ y $z = 24$. Benito selecciona $e = 5$, ya que 5 y 24 no tienen factores comunes. Por último, Benito elige $d = 29$, dado que $5 \cdot 29 - 1$ (es decir, $ed - 1$) es divisible de forma exacta por 24. Benito hace públicos los dos valores, $n = 35$ y $e = 5$, y mantiene en secreto el valor $d = 29$. Observando estos dos valores públicos, supongamos ahora que Alicia quiere enviar a Benito las letras *l*, *o*, *v* y *e*. Interpretando cada letra como un número comprendido entre 1 y 26 (correspondiéndose la *a* con el 1 y la *z* con el 26), Alicia y Benito llevan a cabo las operaciones de cifrado y descifrado mostradas en las Tablas 8.2 y 8.3, respectivamente. Observe que, en este ejemplo, consideramos cada una de las cuatro letras como un mensaje distinto. Un ejemplo más realista sería convertir las cuatro letras en sus representaciones ASCII de 8 bits y luego cifrar el entero correspondiente al patrón de 32 bits resultante. (Tal ejemplo más realista genera números que son demasiado grandes como para imprimirlos aquí.)

Dado que el ejemplo simple de las Tablas 8.2 y 8.3 ha dado lugar a números extremadamente grandes y, puesto que como hemos visto anteriormente, p y q tendrán ambos longitudes de centenares de bits, esto nos lleva a plantearnos varias cuestiones prácticas relativas a RSA: ¿cómo se seleccionan los números primos grandes? ¿Cómo se eligen después e y d? ¿Cómo se lleva a cabo la exponenciación con números grandes? Un análisis de estas importantes cuestiones queda fuera del alcance de este texto, pero el lector puede consultar [Kaufman 1995] y las referencias que incluye para conocer más detalles.

Letra de texto en claro	*m*: representación numérica	m^e	Texto cifrado $c = m^e \bmod n$
l	12	248832	17
o	15	759375	15
v	22	5153632	22
e	5	3125	10

Tabla 8.2 • Cifrado RSA realizado por Alicia, $e = 5$, $n = 35$.

Texto cifrado c	c^d	$m = c^d \bmod n$	Letra de texto en claro
17	481968572106750915091411825223071697	12	l
15	127834039403948858939111232757568359375	15	o
22	851643319086537701956194499721106030592	22	v
10	100000000000000000000000000000	5	e

Tabla 8.3 • Descifrado RSA realizado por Benito, $d = 29$, $n = 35$.

Claves de sesión

Tenemos que resaltar aquí que la exponenciación requerida por RSA es un proceso que consume bastante tiempo. Por el contrario, DES es al menos 100 veces más rápido en software y entre 1.000 y 10.000 veces más rápido en hardware [RSA Fast 2007]. Como resultado, a menudo RSA se utiliza en la práctica en combinación con algún otro mecanismo criptográfico de clave simétrica. Por ejemplo, si Alicia desea enviar a Benito una gran cantidad de datos cifrados podría hacer lo siguiente: primero elegiría una clave para codificar los propios datos; esta clave se conoce como **clave de sesión** y se designa mediante K_S. Alicia tiene que comunicar a Benito la clave de sesión, ya que se trata de la clave simétrica compartida que ambos utilizarán con un sistema de cifrado de clave simétrica (por ejemplo, con DES o AES). Alicia cifra la clave de sesión utilizando la clave pública de Benito, es decir, calcula $c = (K_S)^e \bmod n$. Benito recibe la clave de sesión cifrada mediante RSA, c, y la descifra para obtener la clave de sesión, K_S. De este modo, ahora Benito conoce la clave de sesión que Alicia empleará para transferir los datos cifrados.

¿Por qué funciona RSA?

El cifrado/descifrado de RSA parece algo un poco mágico. ¿Por qué después de aplicar el algoritmo de cifrado y luego el algoritmo de descifrado se recupera el mensaje original? Para comprender por qué funciona RSA, volvamos a fijarnos en la ecuación $n = pq$, donde p y q son los números primos grandes utilizados por el algoritmo RSA.

Recuerde que, con el cifrado RSA, cada mensaje (representado únicamente mediante un entero), m, se eleva a la potencia e utilizando aritmética de módulo n, es decir,

$$c = m^e \bmod n$$

El descifrado se hace elevando este valor a la potencia *d,* utilizando de nuevo aritmética de módulo *n*. El resultado de un paso de cifrado seguido de un paso de descifrado será por tanto $(m^e \bmod n)^d \bmod n$. Veamos ahora qué podemos decir acerca de este valor. Como hemos dicho anteriormente, una propiedad importante de la aritmética modular es que $(a \bmod n)^d \bmod n = a^d \bmod n$ para cualesquiera valores a, n y d. Luego utilizando $a = m^e$ en esta propiedad, tenemos

$$(m^e \bmod n)^d \bmod n = m^{ed} \bmod n$$

Por tanto, queda demostrar que $m^{ed} \bmod n = m$. Aunque estamos intentando eliminar parte de la magia acerca de por qué funciona RSA, para demostrar esta igualdad necesitamos utilizar aquí un resultado que también parece un poco mágico extraído de la teoría de números. Específicamente, necesitamos un resultado que establece que si p y q son primos, y si $n = pq$ y si $z = (p - 1)(q - 1)$, entonces $x^y \bmod n$ es igual a $x^{(y \bmod z)} \bmod n$ [Kaufman 1995]. Aplicando este resultado con $x = m$ e $y = ed$ tenemos

$$m^{ed} \bmod n = m^{(ed \bmod z)} \bmod n$$

Pero recuerde que hemos seleccionado e y d tal que $ed \bmod z = 1$. Esto nos da

$$m^{ed} \bmod n = m^1 \bmod n = m$$

que es exactamente el resultado que estábamos buscando. Elevando en primer lugar a la potencia *e* (es decir, cifrando) y luego elevando a la potencia *d* (es decir, descifrando) obtenemos el valor original, *m.* Aunque *más* maravilloso es el hecho de que si primero elevamos a la potencia *d* y luego a la potencia *e* (es decir, si invertimos el orden del cifrado y el descifrado, realizando en primer lugar la operación de descifrado y aplicando luego la operación de cifrado) también obtenemos el valor original, *m*. Este estupendo resultado se concluye inmediatamente a partir de las reglas de la propia aritmética modular:

$$(m^d \bmod n)^e \bmod n = m^{de} \bmod n = m^{ed} \bmod n = (m^e \bmod n)^d \bmod n$$

La seguridad del algoritmo RSA se basa en el hecho de que no existen algoritmos conocidos para factorizar rápidamente un número, en este caso el valor público *n*, con el fin de obtener los números primos *p* y *q*. Si alguien conociera *p* y *q*, entonces podría calcular fácilmente a partir del valor público *e* la clave secreta *d*. Por otro lado, no se conoce con seguridad si existen o no algoritmos rápidos para factorizar un número, por lo que, en este sentido, la seguridad de RSA no está garantizada.

Otro algoritmo popular de cifrado de clave pública es el algoritmo de Diffie-Hellman, que analizaremos brevemente en los problemas de repaso. El algoritmo de Diffie-Hellman no es tan versátil como RSA, en el sentido de que no puede utilizarse para cifrar mensajes de longitud arbitraria; sin embargo, sí que puede emplearse para establecer una clave de sesión simétrica, que a su vez se utilizará para cifrar los mensajes.

8.3 Integridad de los mensajes y autenticación de los puntos terminales

En la sección anterior hemos visto cómo puede utilizarse el cifrado para proporcionar confidencialidad a dos entidades que desean comunicarse. En esta sección vamos a volver nues-

tra atención al tema criptográfico, igualmente importante, de proporcionar **integridad a los mensajes** (técnica también conocida como autenticación de mensajes). Junto con la integridad de los mensajes, analizaremos dos temas relacionados en esta sección: las firmas digitales y la autenticación de los puntos terminales.

Vamos a definir el problema de la integridad de los mensajes utilizando una vez más a Alicia y a Benito. Suponga que Benito recibe un mensaje (que puede estar cifrado o puede ser texto en claro) y que él cree que este mensaje fue enviado por Alicia. Para autenticar el mensaje, Benito tiene que verificar que:

1. El origen del mensaje es efectivamente Alicia.
2. El mensaje no ha sido alterado mientras viajaba hasta Benito.

Veremos en las Secciones 8.4 a 8.7 que este problema de la integridad de los mensajes es una preocupación crítica en prácticamente todos los protocolos de red seguros.

Como ejemplo específico, considere una red de computadoras en la que se está empleando un algoritmo de enrutamiento de estado del enlace (como por ejemplo OSPF) para determinar las rutas entre cada pareja de routers de la red (véase el Capítulo 4). En un algoritmo de estado del enlace, cada router necesita multidifundir un mensaje de estado del enlace a todos los restantes routers de la red. El mensaje de estado del enlace de un router incluye una lista de su vecinos directamente conectados, junto con los costes directos a esos vecinos. Una vez que un router recibe mensajes de estado de enlace de todos los demás routers puede crear un mapa completo de la red, ejecutar su algoritmo de enrutamiento de coste mínimo y configurar su tabla de reenvío. Un ataque relativamente sencillo contra el algoritmo de enrutamiento consiste en que Tomás distribuya mensajes falsos de estado del enlace con información incorrecta acerca del estado de los enlaces. Debido a la necesidad de integridad de los mensajes, cuando el router B recibe un mensaje de estado del enlace procedente del router A debe verificar que efectivamente el router A ha creado dicho mensaje y, además, que nadie lo ha alterado mientras que el mensaje se encontraba en tránsito.

En esta sección vamos a describir una popular técnica de integridad de mensajes que se utiliza en muchos protocolos de red seguros. Pero, antes de eso, tenemos que tratar otro tema importante dentro del campo de la criptografía: las funciones hash criptográficas.

8.3.1 Funciones hash criptográficas

Como se muestra en la Figura 8.7, una función hash toma una entrada, m, y calcula una cadena de tamaño fijo $H(m)$ conocida con el nombre de hash. La suma de comprobación de Internet (Capítulo 3) y los códigos CRC (Capítulo 4) cumplen con esta definición. Además, una **función hash criptográfica** necesita exhibir la siguiente propiedad adicional:

- Es computacionalmente impracticable encontrar dos mensajes distintos x e y tales que $H(x) = H(y)$.

De una manera informal, podríamos decir que esta propiedad significa que es computacionalmente impracticable que un intruso sustituya un mensaje protegido mediante la función hash por otro mensaje diferente. Es decir, si $(m, H(m))$ son el mensaje y el valor hash de dicho mensaje creado por el emisor, entonces un intruso no puede generar el contenido de otro mensaje, y, que tenga el mismo valor de hash que el mensaje original.

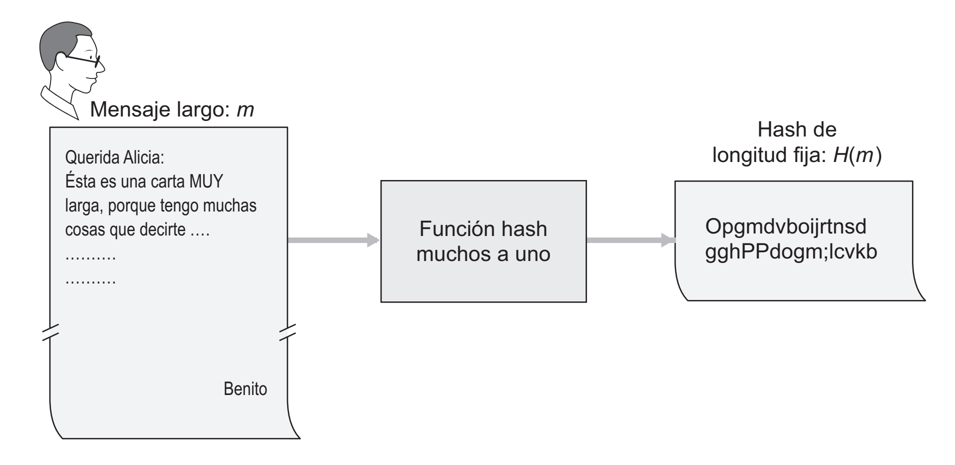


Figura 8.7 • Funciones hash.

Vamos a verificar que una suma de comprobación simple, como la suma de comprobación de Internet, nos proporcionaría una función hash criptográfica bastante poco segura. En lugar de realizar la aritmética en complemento a 1 (como en la suma de comprobación de Internet), calculemos una suma de comprobación tratando cada carácter como un byte y sumando los bytes en fragmentos de 4 bytes cada vez. Suponga que Benito debe a Alicia 100,99 euros y que le envía un mensaje de confirmación que es la cadena de texto "`DEBO100.99BENITO`." La representación ASCII (en notación hexadecimal) de estas letras sería `44, 45, 42, 4F,31, 30, 30, 2E, 39, 39, 42, 45, 4E, 49, 54, 4F`.

La Figura 8.8 (parte superior) muestra que la suma de comprobación de 4 bytes para este mensaje es FC F8 09 11. Un mensaje ligeramente distinto (y mucho más costoso para Benito) es el que se muestra en la parte inferior de esta misma figura. Los mensajes "`DEBO100.99BENITO`" y "`DEBO900.19BENITO`" tienen la *misma* suma de comprobación. Por tanto, este sencillo algoritmo de suma de comprobación violaría el requisito que antes hemos mencionado. Dados los datos originales, es muy sencillo encontrar otro conjunto de datos con la misma suma de comprobación. Obviamente, para propósitos de seguridad necesitaremos una función hash bastante más potente que una mera suma de comprobación.

El algoritmo hash MD5 de Ron Rivest [RFC 1321] se utiliza ampliamente hoy día. Este algoritmo calcula un valor hash de 128 bits mediante un proceso en cuatro pasos, que consiste en un paso de relleno (añadir un uno seguido del número de ceros suficiente como para que la longitud del mensaje satisfaga ciertas condiciones), un paso de agregación (añadir una representación mediante 64 bits de la longitud del mensaje antes de la operación de relleno), una inicialización de un acumulador y un bucle final en el que se procesan los bloques de 16 palabras del mensaje, en cuatro pasadas sucesivas. Consulte [RFC 1321] para ver una descripción de MD5 (incluyendo una implementación con código fuente en C).

El segundo algoritmo principal de hash que se utiliza hoy día es el denominado algoritmo de hash seguro (SHA-1, *Secure Hash Algorithm*) [FIPS 1995]. Este algoritmo está basado en una serie de principios similares a los utilizados en el diseño de MD4 [RFC 1320], el predecesor de MD5. SHA-1, un estándar federal del gobierno de Estados Unidos, es obligatorio siempre que se requiera un algoritmo hash criptográfico para aplicaciones gubernamentales. Este algoritmo produce un resumen del mensaje (*message digest*) de 160 bits. Esa mayor longitud de salida hace que SHA-1 sea más seguro.

Mensaje	Representación ASCII	
D E B O	44 45 42 4F	
1 0 0 .	31 30 30 2E	
9 9 B E	39 39 42 45	
N I T O	4E 49 54 4F	
	FC F8 09 11	Suma de comprobación

Mensaje	Representación ASCII	
D E B O	44 45 42 4F	
9 0 0 .	39 30 30 2E	
1 9 B E	31 39 42 45	
N I T O	4E 49 54 4F	
	FC F8 09 11	Suma de comprobación

Figura 8.8 • El mensaje inicial y el mensaje fraudulento tienen la misma suma de comprobación.

8.3.2 Código de autenticación del mensaje

Volvamos ahora al problema de la integridad de los mensajes. Ahora que sabemos qué son las funciones hash, veamos una primera aproximación de cómo podríamos garantizar la integridad de los mensajes:

1. Alicia crea el mensaje m y calcula el valor hash $H(m)$ (por ejemplo, con SHA-1).
2. Alicia añade a continuación $H(m)$ al mensaje m, creando un mensaje ampliado $(m, H(m))$, el cual envía a Benito.
3. Benito recibe el mensaje ampliado (m, h) y calcula $H(m)$. Si $H(m) = h$, Benito concluye que todo está correcto.

Este enfoque tiene un fallo fundamental. El intruso Tomás puede crear un mensaje ficticio $m´$ en el que dijera que es Alicia, calcular $H(m´)$ y enviar a Benito $(m´, H(m´))$. Cuando Benito recibiera el mensaje, todas las comprobaciones del paso 3 serían correctas, por lo que Benito no sería consciente de que se ha producido un engaño.

Para garantizar la integridad de los mensajes, además de utilizar funciones hash criptográficas Alicia y Benito necesitan un secreto compartido s. Este secreto compartido, que no es más que una cadena de bits, se denomina **clave de autenticación**. Utilizando este secreto compartido puede garantizarse de la forma siguiente la integridad de los mensajes:

1. Alicia crea el mensaje m, concatena s con m para crear $m + s$, y calcula el valor hash $H(m + s)$ (por ejemplo, con SHA-1). $H(m + s)$ se denomina **código de autenticación de mensajes (MAC, *Message Authentication Code*)**.
2. Alicia añade entonces el código MAC al mensaje m, creando un mensaje ampliado $(m, H(m + s))$, y lo envía a Benito.
3. Benito recibe un mensaje ampliado (m, h) y, conociendo s, calcula el valor MAC $H(m + s)$. Si $H(m + s) = h$, Benito concluye que todo está correcto.

En la Figura 8.9 se muestra un resumen de este procedimiento. El lector debe fijarse en que las siglas MAC aquí (que corresponden a "*Message Authentication Code*") no tienen nada que ver con las siglas MAC utilizadas en los protocolos de la capa de enlace (que corresponden a "*Medium Access Control*").

Una característica muy conveniente de los valores MAC es que no se necesita ningún algoritmo de cifrado. De hecho, en muchas aplicaciones, incluyendo el algoritmo de enrutamiento de estado del enlace descrito anteriormente, las entidades que se están comunicando sólo se preocupan de la integridad de los mensajes, mientras que la confidencialidad de los mismos no les importa. Utilizando un código MAC, las entidades pueden autenticar los mensajes que se intercambian sin tener que incluir complejos algoritmos de cifrado en el proceso de garantía de la integridad.

Como cabría esperar, a lo largo de los años se han propuesto diversos estándares para los valores MAC. El estándar más popular hoy día es **HMAC**, que puede utilizarse con MD5 o SHA-1. En la práctica, HMAC hace pasar los datos y la clave de autenticación a través de la función hash dos veces [Kaufman 1995; RFC 2104].

Sigue quedando pendiente una cuestión importante. ¿Cómo distribuimos la clave secreta de autenticación a las distintas entidades que tienen que comunicarse? Por ejemplo, en el algoritmo de enrutamiento de estado del enlace necesitaremos distribuir de alguna manera la clave de autenticación a cada uno de los routers del sistema autónomo. (Observe que todos los routers pueden utilizar la misma clave de autenticación.) Un administrador de red podría llevar a cabo esa distribución visitando físicamente cada uno de los routers. O bien, si el administrador de la red es demasiado perezoso y si cada router tiene su propia clave pública, el administrador puede distribuir la clave de autenticación a cualquiera de los routers cifrándola con la clave pública del router y luego enviando la clave cifrada hasta el router a través de la red.

8.3.3 Firmas digitales

Piense en el número de veces que ha firmado con su nombre en un papel durante la última semana. Todos estamos acostumbrados a firmar cheques, recibos de tarjetas de crédito, documentos legales y cartas. Nuestra firma atestigua el hecho de que nosotros (y no cual-

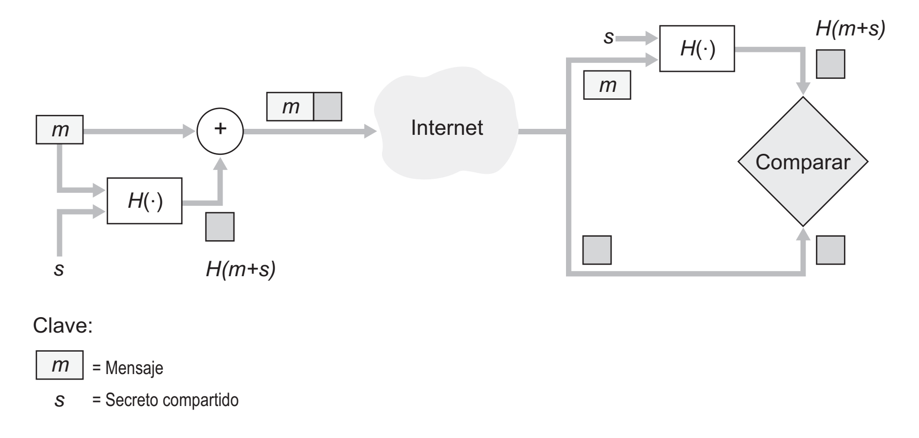


Figura 8.9 • Código de autenticación de mensaje (MAC).

quier otra persona) hemos aceptado y/o acordado el contenido de un documento. En un mundo digital a menudo surge la necesidad de indicar el propietario o creador de un documento o de explicitar nuestro acuerdo con el contenido de un documento. Una **firma digital** es una técnica criptográfica que permite conseguir estos objetivos en el mundo digital.

Al igual que ocurre con las firmas manuscritas, las firmas digitales deben realizarse de forma que sean verificables y no falsificables. Es decir, debe ser posible demostrar que un documento firmado por una persona ha sido, de hecho, firmado por esa persona (la firma tiene que ser verificable) y que *sólo* esa persona podría haber firmado el documento (la firma no puede ser falsificada).

Consideremos ahora cómo podríamos diseñar un esquema de firma digital. Observe que, cuando Benito firma un mensaje, debe poner algo en el mensaje que sea distintivo de él. Benito podría pensar en asociar un valor MAC para la firma, en cuyo caso crearía el valor MAC añadiendo su clave (que permite distinguirle del resto de las personas) al mensaje y luego aplicando la función hash. Pero, para que Alicia pudiera verificar esa firma, también debería disponer de una copia de la clave, en cuyo caso la clave dejaría de ser distintiva de Benito. Por tanto, los códigos MAC no nos permiten conseguir nuestro objetivo en este caso.

Recuerde que, con la criptografía de clave pública, Benito dispone de sendas claves pública y privada, siendo la pareja formada por esas claves distintiva de Benito. Por tanto, la criptografía de clave pública es un candidato excelente para poder proporcionar un mecanismo de firma digital. Veamos ahora cómo se estructura dicho mecanismo.

Suponga que Benito desea firmar digitalmente un documento, *m*. Podemos pensar en el documento como si fuera un archivo o un mensaje que Benito va a firmar y enviar. Como se muestra en la Figura 8.10, para firmar este documento Benito utiliza simplemente su clave privada, K_B^-, para calcular $K_B^-(m)$. De entrada, puede parecer extraño que Benito utilice su clave privada (que, como vimos en la Sección 8.2, se utiliza para descifrar un mensaje que haya sido cifrado con su clave pública) para firmar un documento. Pero recuerde que el cifrado y el descifrado no son otra cosa que operaciones matemáticas (consistentes en elevar a la potencia *e* o *d* en RSA; véase la Sección 8.2) y recuerde también que el objetivo de Benito no es cifrar u ocultar el contenido del documento, sino sólo firmarlo de una manera que sea verificable y no falsificable. La firma digital del documento realizada por Benito es $K_B^-(m)$.

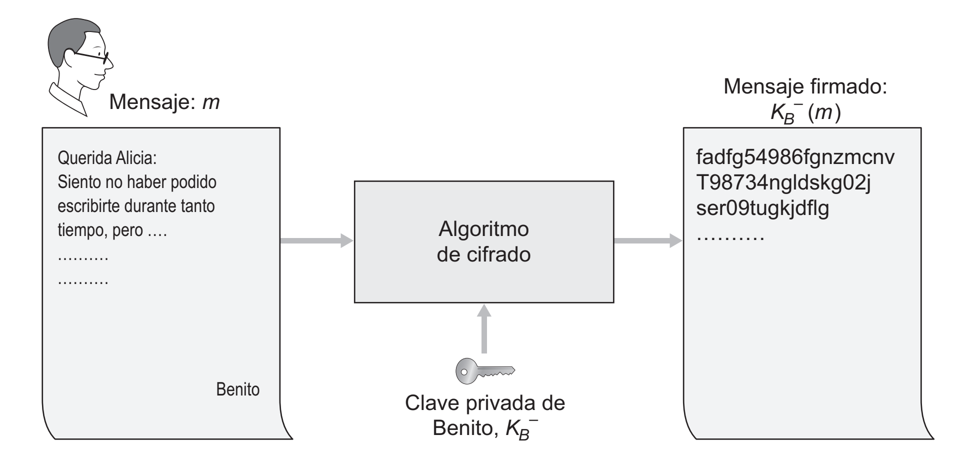


Figura 8.10 • Creación de una firma digital para un documento.

¿Cumple la firma digital $K_B^-(m)$ nuestros requisitos de que sea verificable y no falsificable? Suponga que Alicia dispone de m y $K_B^-(m)$. Ella quiere demostrar ante un tribunal (a consecuencia de algún tipo de litigio) que Benito es quien ha firmado el documento y que es la única persona que podría haberlo firmado. Alicia toma la clave pública de Benito, K_B^+, y se la aplica a la firma digital, $K_B^-(m)$, asociada con el documento, m. Es decir, calcula $K_B^+(K_B^-(m))$, y voilà: con un gesto dramático obtiene m, que se corresponde exactamente con el documento original. Alicia argumenta a continuación que sólo Benito podría haber firmado el documento por las siguientes razones:

- Quienquiera que haya firmado el documento tiene que haber utilizado la clave privada de Benito, K_B^-, a la hora de calcular la firma $K_B^-(m)$ para que $K_B^+(K_B^-(m)) = m$.
- La única persona que puede conocer la clave privada, K_B^-, es el propio Benito. Recuerde, de nuestra exposición sobre RSA de la Sección 8.2, que el conocimiento de la clave pública, K_B^+, no sirve de nada a la hora de determinar la clave privada, K_B^-. Por tanto, la única persona que podría conocer K_B^- es aquella que haya generado la pareja de claves, (K_B^+, K_B^-), es decir, Benito. (Observe que aquí se está suponiendo, sin embargo, que Benito no ha proporcionado su clave privada K_B^- a nadie y que nadie le ha robado K_B^-.)

También es importante observar que si el documento original, m, se modificara de algún modo para obtener una forma alternativa, $m´$, la firma que Benito generara para m no sería válida para $m´$, ya que $K_B^+(K_B^-(m))$ no es igual a $m´$. Por tanto, podemos concluir que las firmas digitales también proporcionan un mecanismo de integridad de los mensajes, permitiendo al receptor verificar que el mensaje no ha sido alterado, además de verificar el origen del mismo.

Uno de los problemas con la firma de datos mediante mecanismos de cifrado es que el cifrado y el descifrado son computacionalmente muy caros. Dada la cantidad adicional de procesamiento que el cifrado y el descifrado exigen, el firmar los datos vía cifrándolos/descifrándolos completamente puede ser como matar moscas a cañonazos. Una técnica más eficiente consiste en introducir funciones hash en el mecanismo de firma digital. Recuerde de la Sección 8.3.2 que un algoritmo hash toma un mensaje, m, de longitud arbitraria y calcula una especie de "huella digital" de longitud fija para el mensaje, que designamos mediante $H(m)$. Utilizando una función hash, Benito firma el valor hash de un mensaje en lugar de firmar el propio mensaje, es decir, Benito calcula $K_B^-(H(m))$. Puesto que $H(m)$ es, generalmente, mucho más pequeño que el mensaje original m, la capacidad de proceso necesario para generar la firma digital se reduce sustancialmente.

En el contexto del envío de un mensaje a Alicia por parte de Benito, la Figura 8.11 proporciona un resumen del procedimiento operativo requerido para crear una firma digital. Benito hace pasar su mensaje original, de gran longitud, a través de una función hash. A continuación, firma digitalmente el valor hash resultante utilizando para ello su clave privada. Después, le envía a Alicia el mensaje original (como texto en claro) junto con el resumen del mensaje digitalmente firmado (al que a partir de ahora denominaremos firma digital). La Figura 8.12 proporciona un resumen del procedimiento operativo de firma. Alicia aplica la clave pública del emisor al mensaje para obtener un valor hash. Asimismo, aplica la función hash al mensaje recibido como texto en claro, para obtener un segundo valor hash. Si los dos valores coinciden, entonces Alicia puede estar segura acerca de la integridad y del autor del mensaje.

Antes de seguir adelante, vamos a comparar brevemente las firmas digitales con los códigos MAC, dado que existen paralelismos entre ambos sistemas, pero también existen

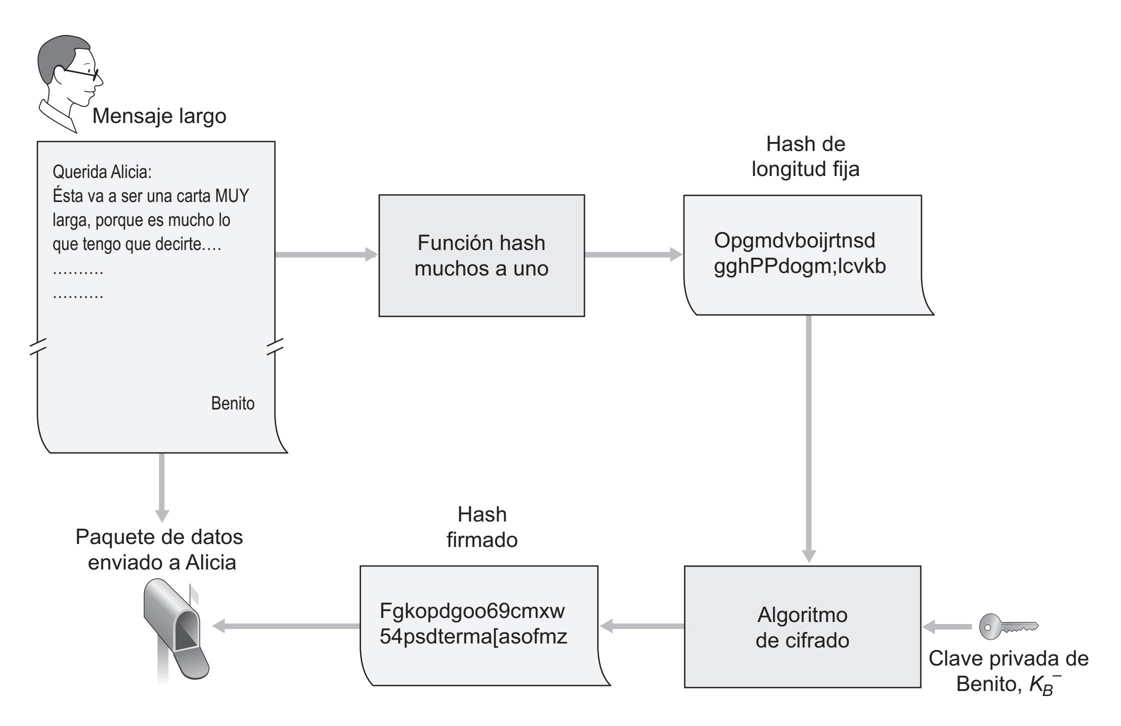


Figura 8.11 • Transmisión de un mensaje firmado digitalmente.

varias diferencias sutiles e importantes. Tanto las firmas digitales como los códigos MAC comienzan con un mensaje (o un documento). Para obtener un valor MAC de ese mensaje, añadimos al mensaje una clave de autenticación y luego calculamos el valor hash del resultado. Observe que, a la hora de crear el valor MAC, no hay involucrado ningún mecanismo de cifrado de clave pública ni de clave simétrica. Para crear una firma digital, primero calculamos el valor hash del mensaje y luego ciframos el mensaje con nuestra clave privada (utilizando criptografía de clave pública). Por tanto, la firma digital es una técnica "más pesada", dado que requiere una Infraestructura de clave pública (PKI, *Public Key Infrastructure*) subyacente, en la que existan autoridades de certificación, como las que más adelante se describen. Veremos en la Sección 8.4 que PGP (un sistema muy popular de correo electrónico seguro) utiliza las firmas digitales para garantizar la integridad de los mensajes. También hemos visto ya que OSPF utiliza valores MAC para garantizar la integridad de los mensajes. En las Secciones 8.5 y 8.6 veremos que los valores MAC también se emplean en varios protocolos de seguridad populares de las capas de transporte y de red.

Certificación de clave pública

Una importante aplicación de las firmas digitales es la **certificación de clave pública**, es decir, certificar que una clave pública pertenece a una entidad específica. Las técnicas de certificación de clave pública se utilizan en muchos protocolos populares de seguridad para las comunicaciones de red, incluyendo IPsec y SSL.

Para tratar de comprender el problema, consideremos una versión de comercio electrónico por Internet del clásico servicio de pizza a domicilio. Alicia trabaja en el sector de la venta de pizza a domicilio y acepta pedidos a través de Internet. Benito, un amante de la

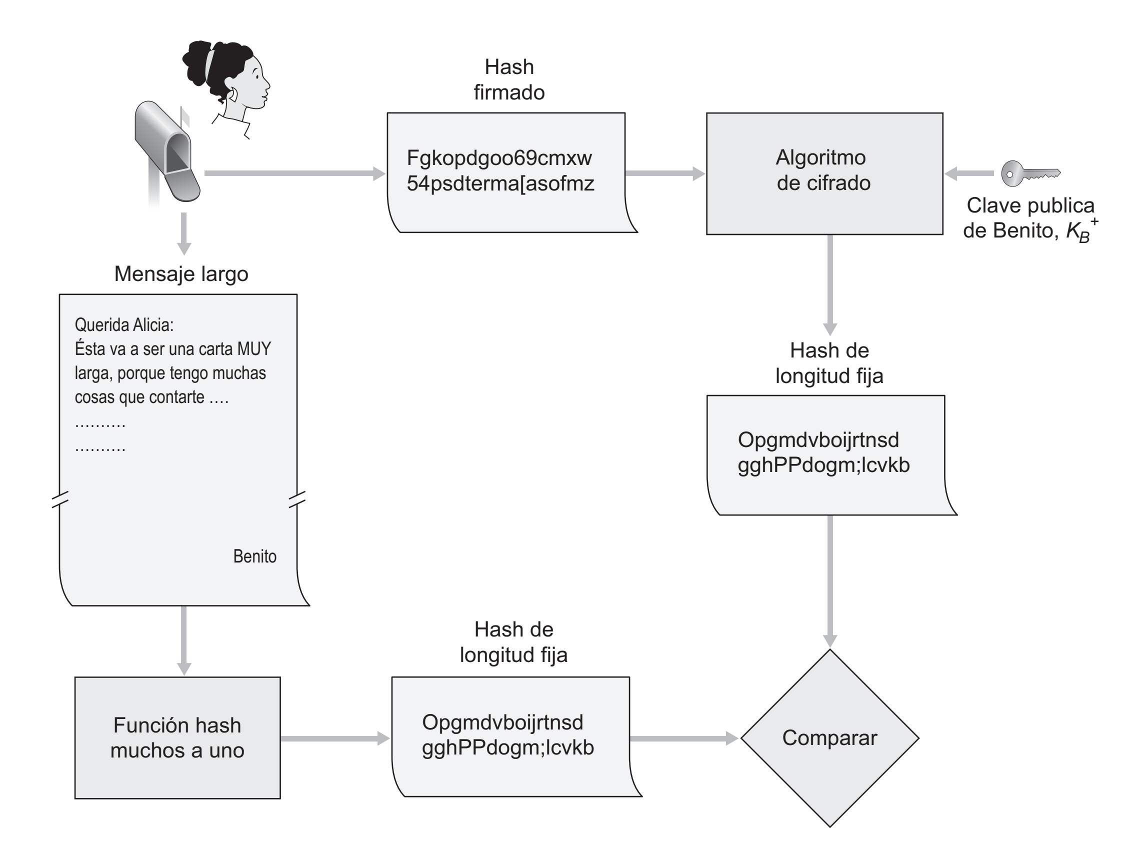


Figura 8.12 • Verificación de un mensaje firmado.

pizza, envía a Alicia un mensaje de texto en claro que incluye su dirección particular y el tipo de pizza que desea. En ese mensaje, Benito también incluye una firma digital (es decir, un valor hash firmado del mensaje en claro original) para demostrar a Alicia que él es el verdadero origen del mensaje. Para verificar la firma, Alicia obtiene la clave de pública de Benito (quizá acudiendo a un servidor de clave pública o a partir del propio mensaje de correo electrónico) y comprueba la firma digital. De esta forma se asegura de que Benito, y no algún adolescente gracioso, ha realizado el pedido.

Este procedimiento parece correcto, hasta que nuestro intruso Tomás entra en escena. Como se muestra en la Figura 8.13, Tomás está tratando de gastar una broma y envía un mensaje a Alicia en el que dice que es Benito, proporciona la dirección particular de Benito y pide una pizza. En este mensaje también incluye su clave pública (la de Tomás), aunque Alicia asume, naturalmente, que se trata de la clave pública de Benito. Tomás también adjunta una firma digital, que habrá creado con su propia clave privada. Después de recibir el mensaje, Alicia aplica la clave pública de Tomás (pensando que es la de Benito) a la firma digital y concluye que el mensaje de texto en claro ha sido creado por Benito. Benito se quedará muy sorprendido cuando el repartidor le lleve a su casa una pizza con pepperoni y anchoas.

Podemos ver a partir de este ejemplo que para que la criptografía de clave pública resulte útil necesitamos poder verificar que disponemos de la verdadera clave pública de la

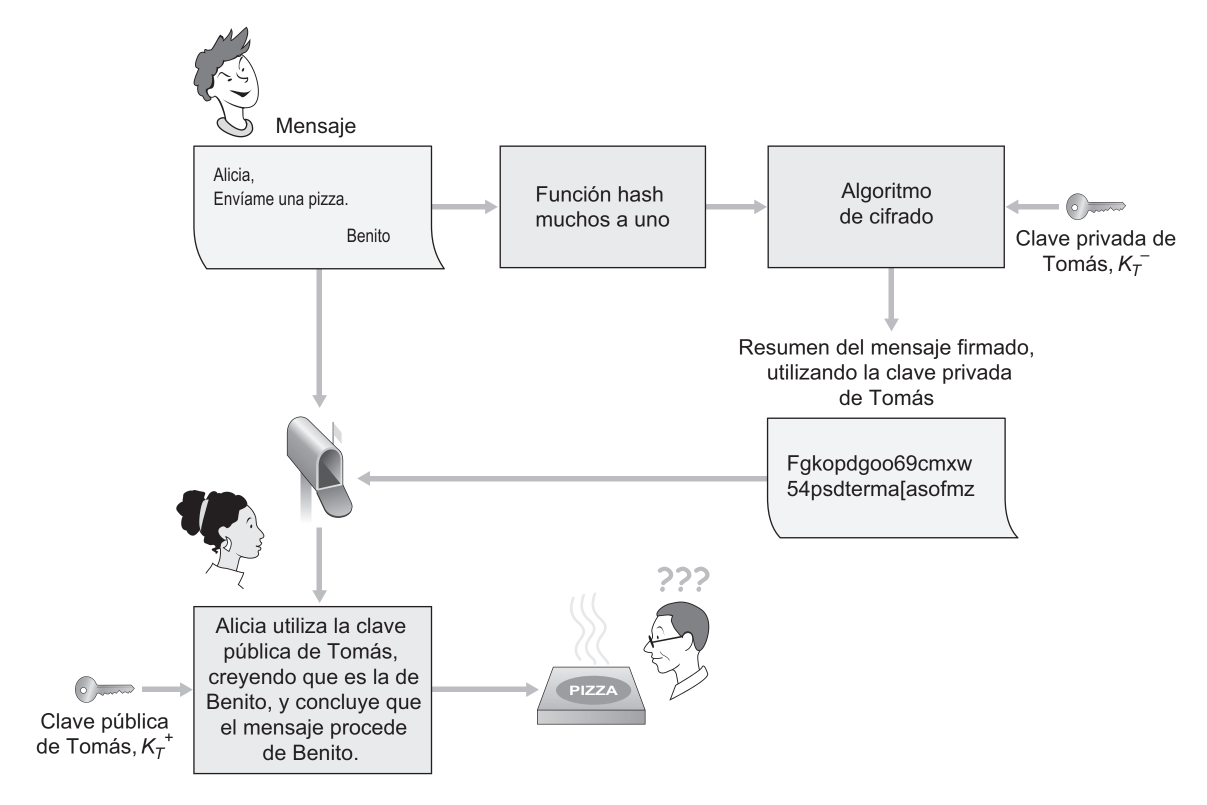


Figure 8.13 • Tomás se hacer pasar por Benito utilizando criptografía de clave pública.

entidad (persona, router, navegador, etc.) con la que queremos comunicarnos. Por ejemplo, cuando Alicia quiere comunicarse con Benito empleando criptografía de clave pública necesita verificar que la clave pública que se supone que pertenece a Benito es verdaderamente de Benito.

La asociación entre una clave pública y una entidad concreta normalmente la realiza una **Autoridad de certificación (CA, *Certification Authority*)**, cuyo trabajo consiste en validar las identidades y emitir los certificados. Una autoridad de certificación desempeña los siguientes papeles:

1. Una CA verifica que una entidad (una persona, un router, etc.) es quien dice ser. No hay ningún procedimiento establecido para la forma en que se lleva a cabo esa certificación. A la hora de tratar con una CA, uno debe confiar en que esa autoridad de certificación habrá realizado una verificación de identidad adecuadamente rigurosa. Por ejemplo, si Tomas fuera capaz de entrar en una autoridad de certificación y anunciar simplemente "Yo soy Alicia" y recibir certificados asociados con la identidad de Alicia, entonces ninguno podríamos tener mucha fe en las claves públicas certificadas por esa autoridad de certificación concreta. Por otro lado, uno tendería (¡o quizá no!) a confiar más en una autoridad de certificación que dependa del gobierno. Sólo podemos confiar en la identidad asociada con una clave pública en la medida en que podamos confiar en una autori-

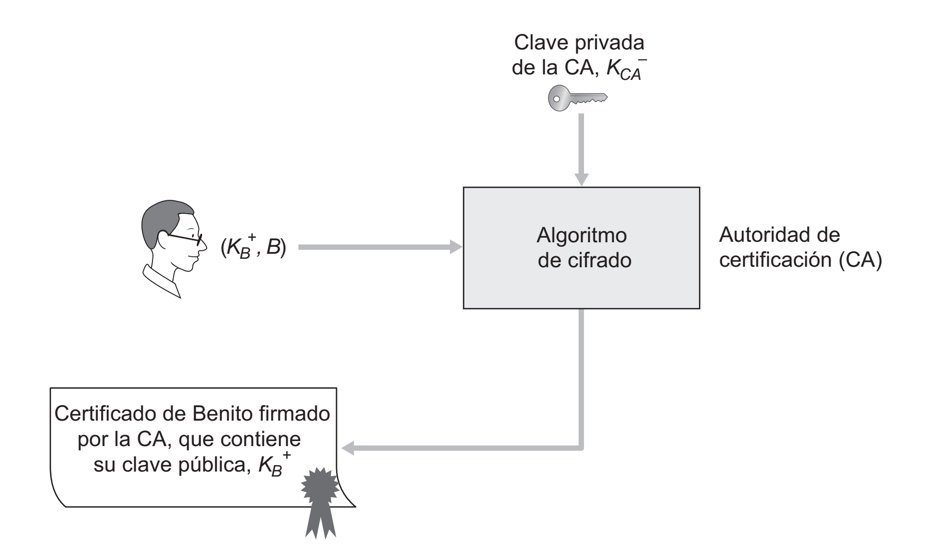


Figura 8.14 • Benito hace que una autoridad de certificación (CA) certifique su clave pública.

dad de certificación y en sus técnicas de verificación de identidad. Como ve, lo que es necesario establecer es una auténtica red de confianza entre distintas entidades.

2. Una vez que la autoridad de certificación verifica la identidad de la entidad, genera un **certificado** que asocia con esa entidad su correspondiente clave pública. El certificado contiene la clave pública y la información de identificación globalmente distintiva acerca del propietario de la clave pública (por ejemplo, el nombre de una persona o una dirección IP). El certificado es firmado digitalmente por la autoridad de certificación. Estos pasos se muestran en la Figura 8.14.

Veamos ahora cómo pueden utilizarse los certificados para combatir a los bromistas de la pizza como Tomás y a otros sujetos indeseables. Cuando Benito hace su pedido, también envía su certificado firmado por la autoridad de certificación. Alicia utiliza la clave pública de la autoridad de certificación para comprobar la validez del certificado de Benito y extraer la clave pública de Benito.

Tanto la Unión Internacional de Telecomunicaciones (ITU, *International Telecommunication Union*) como el IETF han desarrollado estándares para las autoridades de certificación. ITU X.509 [ITU 1993] especifica un servicio de autenticación, así como una sintaxis específica para los certificados. [RFC 1422] describe un mecanismo de administración de claves basado en autoridades de certificación para su utilización con el correo electrónico seguro a través de Internet. Es compatible con X.509, pero va más allá de dicho estándar, al establecer procedimientos y convenios para una arquitectura de gestión de claves. La Tabla 8.4 describe algunos de los campos más importantes de un certificado.

8.3.4 Autenticación del punto terminal

La **autenticación del punto terminal** es el proceso de demostrar a alguien la propia identidad. En la vida cotidiana, las personas nos autenticamos mutuamente de muchas formas

Nombre del campo	Descripción
Versión	Número de versión de la especificación X.509
Número de serie	Identificador único para un certificado emitido por una CA.
Firma	Especifica el algoritmo utilizado por la CA para firmar este certificado.
Nombre del emisor	Identidad de la CA que emite este certificado, en formato de nombre distintivo (DN, *Distinguished Name*) [RFC 2253].
Periodo de validez	Inicio y final del periodo de validez del certificado.
Nombre del sujeto	Identidad de la entidad cuya clave pública está asociada con este certificado, en formato DN.
Clave pública del sujeto	La clave pública del sujeto, así como una indicación del algoritmo de clave pública (y de los parámetros del algoritmo) con el que hay que usar esta clave.

Tabla 8.4 • Campos seleccionados en una clave pública X.509 y RFC 1422.

distintas. Reconocemos la cara de nuestro interlocutor cuando nos lo encontramos; reconocemos nuestras voces a través del teléfono; somos autenticados por los empleados de aduanas que comparan nuestra cara con la fotografía del pasaporte. Pero cuando realizamos una autenticación a través de la red, las dos partes que se comunican no pueden utilizar información biométrica, como la apariencia visual o el timbre de la voz. De hecho, a menudo elementos de red como los routers y los procesos cliente-servidor deben autenticarse mutuamente, basándose exclusivamente en los mensajes y datos intercambiados.

La primera cuestión que debemos plantearnos es si los valores MAC (estudiados en la Sección 8.3.2) pueden utilizarse en la autenticación de los puntos terminales. Supongamos que Alicia y Benito comparten un secreto común, *s*, y que Alicia desea enviar a Benito un mensaje (por ejemplo, un segmento TCP), mientras que Benito quiere asegurarse de que el mensaje recibido procede de Alicia. La técnica natural basada en valores MAC sería la siguiente: Alicia crea un valor MAC utilizando el mensaje y el secreto compartido, añade el valor MAC al mensaje y envía el "mensaje ampliado" resultante a Benito. Cuando éste recibe el mensaje ampliado, utiliza el valor MAC en él contenido para verificar tanto el origen como la integridad del mensaje. De hecho, puesto que sólo Alicia y Benito poseen ese secreto compartido, si los cálculos del valor MAC realizados por Benito proporcionan un resultado que coincide con el valor MAC incluido en el mensaje ampliado, entonces Benito podrá estar seguro de que es Alicia quien ha enviado el mensaje (y de que el mensaje no ha sido alterado durante el trayecto).

Ataques por reproducción y números distintivos

¿Estamos realmente seguros de que Benito puede confiar en que es Alicia quien ha enviado el mensaje? De hecho, Benito no puede estar 100 por cien seguro del origen del mensaje, porque continúa existiendo la posibilidad de que alguien le esté intentando engañar mediante lo que se denomina un **ataque por reproducción**. Como se muestra en la Figura 8.15, Tomás sólo necesita husmear y guardar el mensaje ampliado de Alicia y volver a enviarlo en algún momento posterior. El mensaje repetido podría decir algo así como "Te autorizo a que transfieras un millón de euros de la cuenta de Beato a la de Tomás", lo que haría que se transfiriera un total de dos millones de euros, o el mensaje podría decir "El

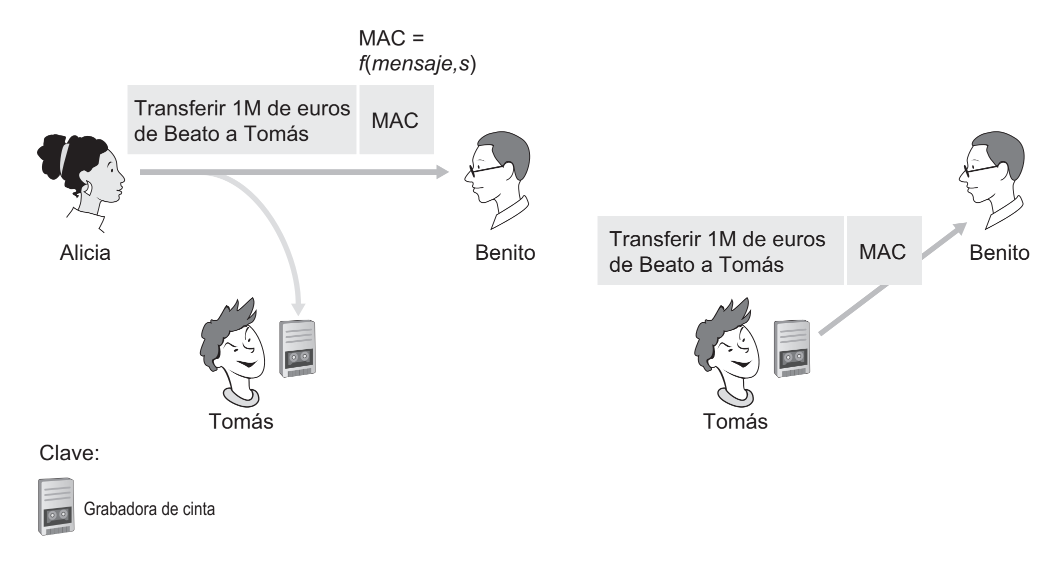


Figura 8.15 • Ataque por reproducción.

enlace desde el router Alicia al router Carlos se ha caído", lo cual, si se envía justo después de que el enlace haya sido reparado, podría provocar la aparición de configuraciones erróneas en las tablas de reenvío.

El escenario de fallo de la Figura 8.15 se produce como resultado del hecho de que Benito no puede distinguir entre el mensaje original enviado por Alicia y la posterior reproducción de ese mensaje original. En otras palabras, Benito no puede estar seguro de si Alicia estaba activa (es decir, se encontraba realmente al otro extremo de la conexión en ese momento) o si el mensaje recibido es una grabación que alguien está reproduciendo. El lector muy (*muy*) atento recordará que el protocolo de acuerdo en tres fases de TCP necesitaba resolver exactamente el mismo problema: el lado de servidor de una conexión TCP no debe aceptar una conexión si el segmento SYN recibido es una copia antigua (retransmisión) de un segmento SYN correspondiente a una conexión anterior. ¿Cómo hacía el servidor TCP para resolver el problema de determinar si el cliente estaba realmente activo? Lo que hacía era seleccionar un número de secuencia inicial que no hubiera sido utilizado durante un periodo de tiempo muy largo, enviar dicho número al cliente y luego esperar a que el cliente respondiera con un segmento ACK que contuviera dicho número. Podemos adoptar aquí la misma idea con propósitos de autenticación.

Un **número distintivo** (*nonce, number used once*) es un número que un protocolo sólo empleará una vez en toda su vida. Es decir, una vez que un protocolo utiliza un número distintivo, jamás volverá a usar dicho número. Como se muestra en la Figura 8.16, nuestro nuevo protocolo emplea el número distintivo de la forma siguiente:

1. Benito selecciona un número distintivo, *R,* y se lo envía a Alicia. Observe que el número distintivo se envía en forma de texto legible. Alicia crea entonces el valor MAC utilizando su mensaje original, el secreto compartido *s* y el número distintivo *R.* (Por ejemplo, para crear el valor MAC Alicia puede concatenar tanto el secreto compartido como el número distintivo con el mensaje y pasar el resultado a través de una función hash.) Alicia añade el valor MAC al mensaje, crea un mensaje ampliado y envía dicho mensaje a Benito.

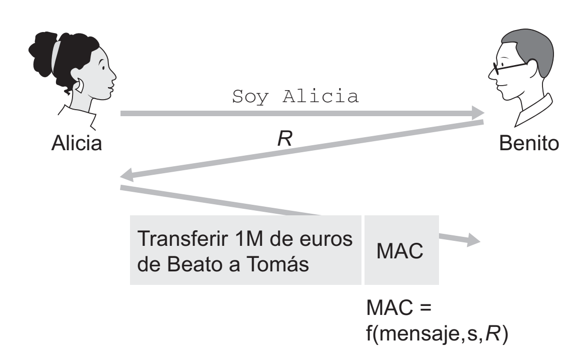


Figura 8.16 • Defensa frente a los ataques por reproducción utilizando números distintivos.

2. Benito calcula un valor MAC a partir del mensaje (contenido en el mensaje ampliado), del número distintivo R y del secreto compartido s. Si el valor MAC resultante es igual al valor MAC contenido en el mensaje ampliado, Benito no sólo sabrá que Alicia ha generado el mensaje, sino también que lo ha generado *después* de que Benito enviara el número distintivo, dado que el valor de ese número distintivo es necesario para calcular el valor MAC correcto.

Como veremos en las Secciones 8.5 y 8.6 al hablar de SSL e IPsec, la combinación de números distintivos y valores MAC se utiliza a menudo en protocolos de red seguros para proporcionar tanto integridad de los mensajes como autenticación del punto terminal.

Pero, ¿qué pasa si Alicia desea enviar una serie de mensajes, por ejemplo, una serie de segmentos TCP? ¿Tendrá Benito que enviar a Alicia un nuevo número distintivo por cada uno de los mensajes? En la práctica, sólo hace falta un único número distintivo. Como veremos en la Sección 8.5 cuando hablemos de SSL, un único número distintivo combinado con números de secuencia permitirá a Benito verificar que todos los mensajes procedentes de Alicia son recientes.

Autenticación mediante criptografía de clave pública

El uso de un número distintivo y un secreto compartido permite definir un protocolo de autenticación perfectamente adecuado. Una pregunta natural es si podemos emplear un número distintivo y la criptografía de clave pública para resolver también el problema de la autenticación. La utilización de una técnica basada en clave pública permitiría resolver una dificultad de cualquier sistema basado en un secreto compartido: nos referimos a la dificultad de tener que preocuparse por el modo en que ambos interlocutores pueden llegar a conocer el valor del secreto compartido. Un protocolo natural basado en la criptografía de clave pública sería el siguiente:

1. Alicia envía a Benito el mensaje "`Soy Alicia`".
2. Benito selecciona un número distintivo, R, y se lo envía a Alicia. Una vez más, el número distintivo se emplea para garantizar que Alicia está activa.
3. Alicia utiliza su *clave privada*, K_A^-, para cifrar el número distintivo y envía el valor resultante, $K_A^-(R)$, a Benito. Puesto que sólo Alicia conoce su clave privada, nadie excepto Alicia podrá generar $K_A^-(R)$.

4. Benito aplica la clave pública de Alicia, K_A^+, al mensaje recibido; es decir, Benito calcula $K_A^+(K_A^-(R))$. Recuerde de nuestra exposición sobre la criptografía de clave pública con RSA de la Sección 8.2 que $K_A^+(K_A^-(R)) = R$. De este modo, Benito calcula R y autentica a Alicia.

La operación de este protocolo se ilustra en la Figura 8.17. ¿Es seguro este protocolo? Puesto que utiliza técnicas de clave pública, requiere que Benito consulte la clave pública de Alicia. Esto conduce a un escenario bastante interesante, mostrado en la Figura 8.18, en el que Tomás puede ser capaz de hacerse pasar por Alicia ante Benito.

1. Tomás envía a Benito el mensaje “`Soy Alicia`”.
2. Benito selecciona un número distintivo, R, y se lo envía a Alicia, pero el mensaje es interceptado por Tomás.
3. Tomás emplea su clave privada, K_T^-, para cifrar el número distintivo y envía el valor resultante, $K_T^-(R)$, a Benito. Para Benito, $K_T^-(R)$ es simplemente una serie de bits y no sabe si esos bits representan $K_T^-(R)$ o $K_A^-(R)$.
4. Benito debe ahora obtener la clave pública de Alicia para poder aplicar K_A^+ al valor que acaba de recibir. Envía un mensaje a Alicia preguntándola cuál es su clave pública K_A^+ (Benito también podría obtener la clave pública de Alicia de su sitio web). Tomás intercepta también este mensaje y responde a Benito con K_T^+, es decir, con la clave pública de Tomás. Benito calcula $K_T^+ (K_T^- (R)) = R$ y, de ese modo, autentica a Tomás como si fuera Alicia.

Teniendo en cuenta este escenario, es evidente que la seguridad de este protocolo de autenticación de clave pública depende completamente de la seguridad del sistema de distribución de claves públicas. Afortunadamente, podemos utilizar certificados para distribuir claves públicas de forma segura, como vimos en la Sección 8.3.

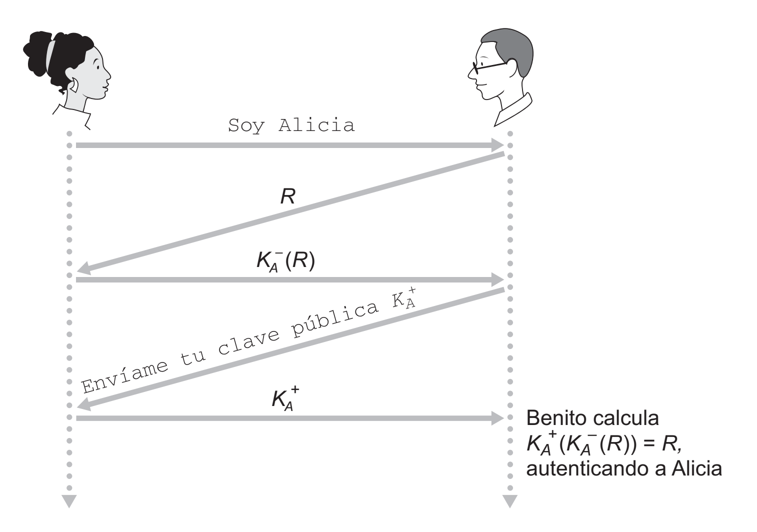


Figura 8.17 • Protocolo de autenticación de clave pública que funciona correctamente.

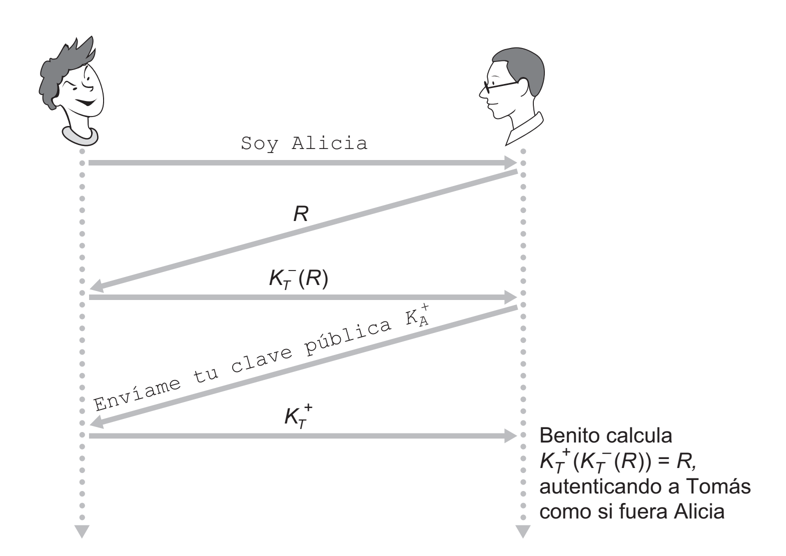


Figura 8.18 • Un agujero de seguridad en el protocolo de autenticación de clave pública.

En el escenario de la Figura 8.18 Benito y Alicia pueden llegar a descubrir que hay algo que va mal en cuanto Benito afirme haber interactuado con Alicia, mientras que Alicia sabe que nunca ha interactuado con Benito. Pero existe un ataque todavía más insidioso que evitaría incluso esta detección. En el escenario de la Figura 8.19 Alicia y Benito están hablando entre ellos, pero explotando el mismo agujero de seguridad en el protocolo de autenticación, Tomás es capaz de interponerse *de forma transparente* entre Alicia y Benito. En particular, si Benito comienza a enviar datos cifrados a Alicia utilizando la clave de cifrado que ha recibido de Tomás, Tomás puede extraer el texto en claro correspondiente a la comunicación entre Benito y Alicia. Al mismo tiempo, Tomás puede reenviar los datos de Benito a Alicia (después de volver a cifrarlos utilizando la clave pública real de Alicia).

Benito está convencido de estar enviando datos cifrados y Alicia está convencida de estar recibiendo datos cifrados utilizando su propia clave pública; ambos son inconscientes de la presencia de Tomás. Si Benito y Alicia se reunieran posteriormente y hablaran de la interacción que han tenido, Alicia habría recibido exactamente lo que Benito envió, por lo que ninguno de ellos detectaría que algo va mal. Esto es un ejemplo del denominado **ataque por interposición (*man-in-the-middle attack*)**.

8.4 Correo electrónico seguro

En las secciones anteriores hemos examinado una serie de problemas fundamentales en el campo de la seguridad de red, incluyendo las técnicas de criptografía de clave simétrica y de clave pública, las de autenticación de punto terminal, los mecanismos de distribución de claves, el problema de la integridad de los mensajes y las técnicas de firma digital. Ahora vamos a examinar cómo se utilizan hoy en día dichas herramientas para proporcionar seguridad en Internet.

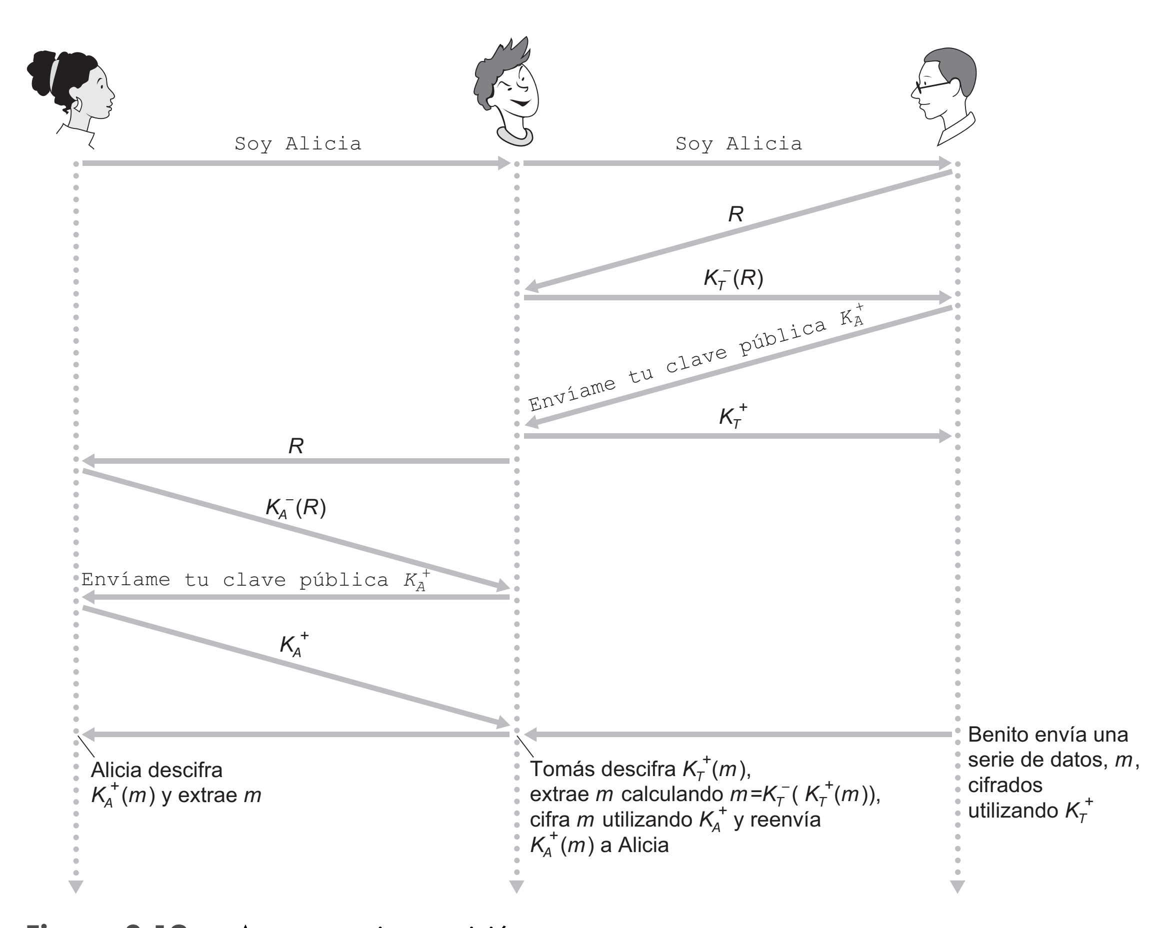


Figura 8.19 • Ataque por interposición.

Es interesante observar que es posible proporcionar servicios de seguridad en cualquiera de las cuatro capas superiores de la pila de protocolos de Internet. Cuando se proporciona seguridad para un protocolo específico de la capa de aplicación, la aplicación que utiliza ese protocolo disfrutará de uno o más servicios de seguridad, como los de confidencialidad, autenticación o integridad. Cuando los mecanismos de seguridad se proporcionan para un protocolo de la capa de transporte, todas las aplicaciones que usan dicho protocolo disfrutarán de los servicios de seguridad del protocolo de transporte. Cuando la seguridad se proporciona en la capa de red en un esquema host a host, todos los segmentos de la capa de transporte (y por tanto todos los datos de la capa de aplicación) disfrutarán de los servicios de seguridad de la capa de red. Cuando se proporciona seguridad a nivel de enlace, entonces los datos de todas las tramas que viajan a través del enlace utilizarán los servicios de seguridad del enlace.

En las Secciones 8.4 a 8.7 vamos a examinar el modo en que se utilizan las técnicas de seguridad en las capas de aplicación, transporte, red y de enlace de datos. Para ser coherentes con la estructura general del libro, comenzaremos por la parte superior de la pila de protocolos, abordando la seguridad en la capa de aplicación. Nuestro enfoque consiste en utilizar una aplicación específica, como el correo electrónico, como caso de estudio de las

técnicas de seguridad de la capa de aplicación. Después descenderemos por la pila de protocolos y examinaremos el protocolo SSL (que proporciona seguridad a la capa de transporte), IPsec (que proporciona seguridad en la capa de red) y los mecanismos de seguridad del protocolo IEEE 802.11 para redes LAN inalámbricas.

El lector puede estarse preguntando por qué se proporciona la funcionalidad de seguridad en más de una capa dentro de Internet. ¿No bastaría simplemente con proporcionar la funcionalidad de seguridad en la capa de red? Existen dos respuestas a esta pregunta. En primer lugar, aunque la seguridad en la capa de red puede proporcionar la funcionalidad básica de cifrado de todos los datos contenidos en los datagramas (es decir, de todos los segmentos de la capa de transporte) y de autenticar todas las direcciones IP de origen, lo que no puede es ofrecer seguridad de nivel de usuario. Por ejemplo, un sitio web de comercio electrónico no puede confiar en la seguridad de la capa IP para autenticar a un cliente que esté comprando bienes o servicios en ese sitio. Por tanto, existe una necesidad de incorporar funcionalidad de seguridad en las capas superiores, además de la funcionalidad básica en las capas inferiores. En segundo lugar, es más fácil generalmente implantar servicios Internet nuevos, incluyendo los servicios de seguridad, en las capas superiores de la pila de protocolos. Mientras se espera a que un nuevo mecanismo de seguridad se implante de forma generalizada en la capa de red, lo que puede llegar a tardar años, muchos desarrolladores de aplicaciones simplemente se ponen manos a la obra e introducen la funcionalidad de seguridad en sus aplicaciones favoritas. Un ejemplo clásico es PGP (*Pretty Good Privacy*), que proporciona correo electrónico seguro (y de la que hablaremos posteriormente en esta sección). Requiriendo únicamente código de aplicación de cliente y de servidor, PGP fue una de las primeras tecnologías de seguridad que se utilizó ampliamente en Internet.

8.4.1 Correo electrónico seguro

Ahora vamos a utilizar los principios criptográficos de las Secciones 8.2 a 8.3 para crear un sistema de correo electrónico seguro. Crearemos este diseño de alto nivel de una forma incremental, introduciendo en cada paso nuevos servicios de seguridad. A la hora de diseñar un sistema de correo electrónico seguro debemos tener presente el ejemplo visto en la Sección 8.1: la relación amorosa entre Alicia y Benito. Imagine que Alicia quiere enviar un mensaje de correo electrónico a Benito y que Tomás pretende inmiscuirse en la conversación.

Antes de seguir adelante y diseñar un sistema de correo electrónico seguro para Alicia y Benito, tenemos que considerar qué características de seguridad serían más deseables para ellos. La primera y principal es la *confidencialidad.* Como hemos explicado en la Sección 8.1, ni Alicia ni Benito quieren que Tomás lea el correo electrónico de Alicia. La segunda característica que Alicia y Benito probablemente deseen para su sistema de correo electrónico seguro es la de *autenticación del emisor*. En particular, cuando Benito recibe el mensaje "`Ya no te quiero. No quiero verte nunca más. Olvídame, Alicia.`" naturalmente le gustaría estar seguro de que el mensaje procede de Alicia y no de un intruso. Otra característica que los dos amantes apreciarían sería la de *integridad de los mensajes*, es decir, una garantía de que los mensajes que Alicia envíe no sean modificados a lo largo del camino hasta llegar a Benito. Por último, el sistema de correo electrónico debe proporcionar una *autenticación del receptor*; es decir, Alicia querrá asegurarse de que está enviando su mensaje a Benito y no a alguna otra persona (por ejemplo, Tomás) que esté haciéndose pasar por Benito.

Así que comencemos analizando el primero de los objetivos, la confidencialidad. La forma más directa de proporcionar confidencialidad es que Alicia cifre el mensaje con una tecnología de clave simétrica (como DES o AES) y que Benito descifre el mensaje al recibirlo. Como se explica en la Sección 8.2, si la clave simétrica es lo suficiente larga y si únicamente Alicia y Benito tienen la clave, entonces es extremadamente difícil que ninguna otra persona (incluyendo a Tomás) lea el mensaje. Aunque esta técnica es muy sencilla, presenta la dificultad fundamental de la que ya hemos hablado en la Sección 8.2: distribuir una clave simétrica de modo que sólo Alicia y Benito tengan copia de la misma. Por tanto, vamos a considerar una solución alternativa: la criptografía de clave pública (utilizando por ejemplo RSA). En la técnica de clave pública, Benito pone a disposición de todo el mundo su clave pública (por ejemplo, en un servidor de claves públicas o en su página web personal). Alicia cifra su mensaje con la clave pública de Benito y envía el mensaje cifrado a la dirección de correo electrónico de Benito. Cuando éste recibe el mensaje, simplemente lo descifra con su clave privada. Suponiendo que Alicia esté segura de que la clave pública utilizada es la de Benito, esta técnica constituye un método excelente para proporcionar la confidencialidad deseada. Sin embargo, un problema es que el cifrado de clave pública es relativamente poco eficiente, particularmente para mensajes de gran longitud.

Para solventar este problema de eficiencia, vamos a hacer uso de una clave de sesión (de lo que hemos hablado en la Sección 8.2.2). En particular, Alicia (1) selecciona una clave de sesión simétrica aleatoria, K_S, (2) cifra su mensaje, *m,* con la clave simétrica, (3) cifra la clave simétrica con la clave pública de Benito, K_B^+, (4) concatena el mensaje cifrado y la clave simétrica cifrada para formar un "paquete" y (5) envía este paquete a la dirección de correo electrónico de Benito. Los pasos se ilustran en la Figura 8.20. (En ésta y las subsiguientes figuras, el símbolo "+" encerrado en un círculo representa la operación de concatenación y el signo "–" encerrado en un círculo representa la operación de desconcatenación.) Cuando Benito recibe el paquete, (1) utiliza su clave privada, K_B^-, para obtener la clave simétrica, K_S, y (2) utiliza la clave simétrica K_S para descifrar el mensaje *m*.

Habiendo diseñado un sistema de correo electrónico seguro que proporciona un servicio de confidencialidad, vamos a diseñar ahora otro sistema que proporcione tanto autenticación del emisor como integridad de los mensajes. Vamos a suponer por el momento que a Alicia y Benito ya no les preocupa la confidencialidad (¡quieren compartir sus sentimientos con todo el mundo!) y que sólo les preocupa la autenticación del emisor y la integridad

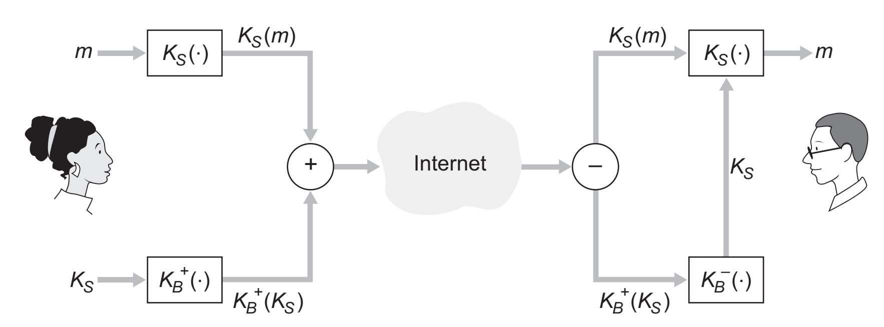


Figura 8.20 • Alicia utiliza una clave de sesión simétrica, K_S, para enviar un mensaje secreto de correo electrónico a Benito.

de los mensajes. Para llevar a cabo esta tarea, utilizaremos firmas digitales y resúmenes de mensajes, como se describe en la Sección 8.3. Específicamente, Alicia (1) aplica una función hash *H* (por ejemplo, MD5) a su mensaje, *m,* para obtener un resumen del mensaje, (2) firma el resultado de la función hash con su clave privada, K_A^-, para crear una firma digital, (3) concatena el mensaje original (no cifrado) con la firma para crear un paquete y (4) envía el paquete a la dirección de correo electrónico de Benito. Cuando éste recibe el paquete, (1) aplica la clave pública de Alicia, K_A^+, al resumen del mensaje firmado y (2) compara el resultado de esta operación con su propio valor hash, *H,* del mensaje. Estos pasos se ilustran en la Figura 8.21. Como se ha estudiado en la Sección 8.3, si los dos resultados coinciden, Benito puede estar seguro de que el mensaje procede de Alicia y no ha sido alterado.

Vamos a considerar ahora el diseño de un sistema de correo electrónico que proporcione confidencialidad, autenticación del emisor *e* integridad de los mensajes. Esto puede hacerse combinando los procedimientos de las Figuras 8.20 y 8.21. Alicia crea primero un paquete preliminar, exactamente como en la Figura 8.21, que está compuesto por su mensaje original y un valor hash digitalmente firmado del mensaje. A continuación, trata este paquete preliminar como un mensaje en sí mismo y envía este nuevo mensaje a través de los pasos del emisor de la Figura 8.20, creando un nuevo paquete que es enviado a Benito. Los pasos aplicados por Alicia se muestran en la Figura 8.22. Cuando Benito recibe el paquete, primero aplica su lado correspondiente de la Figura 8.20 y luego su lado correspondiente de la Figura 8.21. Debería quedar claro para el lector que este diseño permite conseguir el objetivo de proporcionar confidencialidad, autenticación del emisor e integridad de los mensajes. Observe que, con este esquema, Alicia utiliza la criptografía de clave pública dos veces: una vez con su propia clave privada y otra con la clave pública de Benito. De forma similar, Benito también emplea la criptografía de clave pública dos veces: una con su clave privada y otra con la clave pública de Alicia.

El diseño de correo electrónico seguro esbozado en la Figura 8.22 probablemente proporciona una seguridad satisfactoria para la mayoría de los usuarios de correo electrónico, en la mayoría de las ocasiones. Pero continuamos teniendo un problema importante que hay que resolver. El diseño de la Figura 8.22 requiere que Alicia obtenga la clave pública de Benito y que Benito obtenga la clave pública de Alicia. La distribución de estas claves

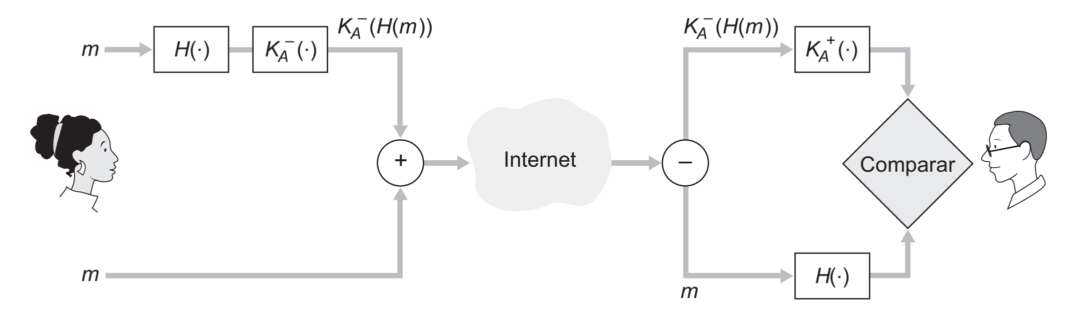


Figura 8.21 • Utilización de funciones hash y firmas digitales para proporcionar autenticación del emisor e integridad de los mensajes.

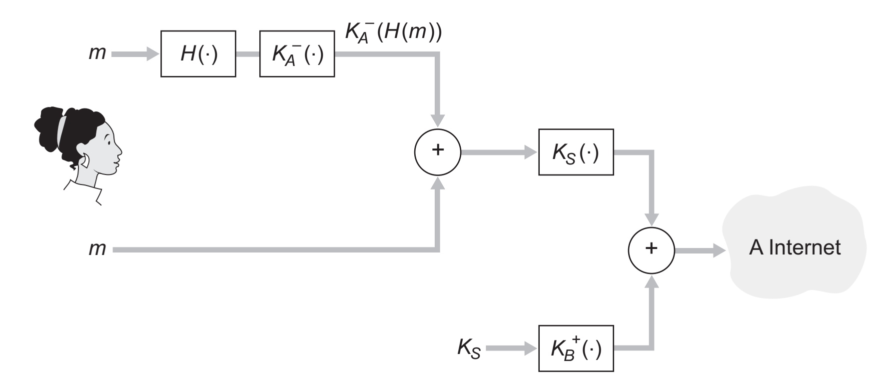


Figura 8.22 • Alicia utiliza la criptografía de clave simétrica, la criptografía de clave pública, una función hash y una firma digital para obtener los servicios de confidencialidad, autenticación del emisor e integridad de los mensajes.

públicas no es un problema trivial. Por ejemplo, Tomás podría hacerse pasar por Benito y darle a Alicia su propia clave pública, mientras que le hace creer que se trata de la clave pública de Benito, lo que permitiría a Tomás recibir el mensaje destinado a Benito. Como hemos visto en la Sección 8.3, una técnica popular para distribuir de forma segura las claves públicas consiste en *certificar* esas claves públicas utilizando una autoridad de certificación (CA).

8.4.2 PGP

Escrito por Phil Zimmermann en 1991, **Pretty Good Privacy (PGP)** es un esquema de cifrado de correo electrónico que se ha convertido en un estándar *de facto*. Su sitio web sirve más de un millón de páginas al mes a usuarios de 166 países distintos [PGPI 2007]. Hay versiones de PGP disponibles en el dominio público; por ejemplo, podemos encontrar el software PGP para nuestra plataforma favorita, así como una gran cantidad de artículos interesantes en la página web internacional de PGP [PGPI 2007]. (Un ensayo particularmente interesante escrito por el autor de PGP es [Zimmermann 2007].) Básicamente, el diseño de PGP es igual al mostrado en la Figura 8.22. Dependiendo de la versión, el software PGP utiliza MD5 o SHA para calcular el resumen del mensaje; CAST, triple-DES o IDEA para el cifrado de clave simétrica y RSA para el cifrado de clave pública.

Cuando se instala PGP, el software crea una pareja de clave pública y clave privada para el usuario. La clave pública puede darse a conocer a todo el mundo a través del sitio web del usuario o puede almacenarse en un servidor de clave pública. La clave privada está protegida mediante el uso de una contraseña. La contraseña debe introducirse cada vez que el usuario accede a la clave privada. PGP le da al usuario la opción de firmar digitalmente el mensaje, de cifrar el mensaje o de realizar tanto la operación de firma como la de cifrado. La Figura 8.23 muestra un mensaje firmado PGP. Este mensaje aparece después de la cabecera MIME. Los datos codificados en el mensaje son iguales a $K_A^-(H(m))$, es decir, son un resumen firmado digitalmente del mensaje. Como hemos comentado anteriormente, para que Benito verifique la integridad del mensaje necesitará tener acceso a la clave pública de Alicia.



PHIL ZIMMERMANN Y PGP

Philip R. Zimmermann es el creador de Pretty Good Privacy (PGP). Debido a ello fue objeto de una investigación criminal que duró tres años, porque el gobierno americano sostenía que las restricciones de exportación de los Estados Unidos para el software criptográfico fueron violadas cuando PGP se difundió por todo el mundo después de su publicación en 1991 como software gratuito. Después de distribuir PGP como shareware, alguna otra persona lo puso en Internet y los ciudadanos extranjeros pudieron descargárselo. Los programas criptográficos en Estados Unidos están clasificados como las municiones según las leyes federales y no se pueden exportar.

A pesar de la falta de financiación, la falta de personal contratado y la falta de una empresa que le apoyara y a pesar de las intervenciones gubernamentales, PGP llegó de todos modos a convertirse en el software de cifrado de correo electrónico más ampliamente utilizado en todo el mundo. Resulta irónico que el propio gobierno de Estados Unidos pueda haber contribuido inadvertidamente a la difusión de PGP a causa del caso Zimmermann.

El gobierno de Estados Unidos abandonó el caso a principios de 1996. El anuncio fue enormemente celebrado por los activistas de Internet. El caso Zimmermann se había convertido en la historia de una persona inocente luchando por sus derechos contra los abusos de un gobierno fuerte. La rendición del gobierno fue muy bien recibida, en parte debido a la campaña en favor de la censura en Internet que estaba teniendo lugar en el Congreso americano y a las presiones del FBI para que se aumentaran las facultades de intercepción de las comunicaciones concedidas al gobierno.

Después de que el gobierno abandonara el caso, Zimmermann fundó PGP Inc., que fue adquirida por Network Associates en diciembre de 1997. Zimmermann es ahora un consultor independiente en el campo de la criptografía.

La Figura 8.24 muestra un mensaje PGP secreto. Este mensaje también aparece después de la cabecera MIME. Por supuesto, el mensaje de texto en claro no está incluido en el mensaje secreto de correo electrónico. Cuando un emisor (como Alicia) quiere tanto confidencialidad como integridad, PGP contiene un mensaje como el de la Figura 8.24 dentro del mensaje de la Figura 8.23.

```
-----BEGIN PGP SIGNED MESSAGE-----
Hash:  SHA1
Benito:
¿Puedo verte esta noche?
Apasionadamente tuya, Alicia
-----BEGIN PGP SIGNATURE-----
Version: PGP for Personal Privacy 5.0
Charset:  noconv
yhHJRHhGJGhgg/12EpJ+lo8gE4vB3mqJhFEvZP9t6n7G6m5Gw2
-----END PGP SIGNATURE-----
```

Figura 8.23 • Un mensaje firmado mediante PGP.

```
-----BEGIN PGP MESSAGE-----
Version: PGP for Personal Privacy 5.0
u2R4d+/jKmn8Bc5+hgDsqAewsDfrGdszX68liKm5F6Gc4sDfcXyt
RfdS10juHgbcfDssWe7/K=lKhnMikLo0+1/BvcX4t==Ujk9PbcD4
Thdf2awQfgHbnmKlok8iy6gThlp
-----END PGP MESSAGE
```

Figura 8.24 • Un mensaje PGP secreto.

PGP también proporciona un mecanismo para la certificación de clave pública, pero el mecanismo es bastante distinto del sistema más convencional basado en autoridades de certificación. Las claves públicas PGP están certificadas por una *red de confianza.* La propia Alicia puede certificar cualquier pareja de clave/nombre de usuario cuando crea que esa pareja es realmente correcta. Además, PGP permite que Alicia diga que confía en algún otro usuario a la hora de certificar la autenticidad de otras claves. Algunos otros usuarios PGP firman mutuamente sus claves manteniendo reuniones específicas de firma de claves. Los usuarios se reúnen físicamente, intercambian sus claves públicas y certifican las claves unos con otros firmándolas con sus claves privadas.

8.5 Conexiones TCP seguras: SSL

En la sección anterior hemos visto cómo las técnicas criptográficas pueden proporcionar confidencialidad, integridad de los datos y autenticación del punto terminal a una aplicación específica, que en nuestro caso concreto era el correo electrónico. En esta sección vamos a descender un nivel dentro de la pila de protocolos para examinar cómo puede utilizarse la criptografía para mejorar los servicios de seguridad de TCP, incluyendo la confidencialidad, la integridad de los datos y la autenticación del punto terminal. Esta versión mejorada de TCP se conoce comúnmente como **Capa de sockets seguros (SSL, *Secure Sockets Layer*)**. Una versión ligeramente modificada de SSL versión 3, denominada **Seguridad de la capa de transporte (TLS, *Transport Layer Security*)**, ha sido estandarizada por el IETF [RFC 2246].

SSL fue diseñado originalmente por Netscape, pero las ideas básicas subyacentes a dotar de seguridad a TCP han terminado evolucionando a partir de los trabajos de Netscape (consulte, por ejemplo, Woo [Woo 1994]). Desde su invención, SSL ha disfrutado de una amplia implantación. SSL está soportado por todos los navegadores y servidores web más populares y es empleado por la práctica totalidad de los sitios de comercio electrónico de Internet (incluyendo Amazon, eBay, Yahoo!, MSN, etc.). Anualmente se gastan decenas de miles de millones de euros a través de SSL. De hecho, si el lector ha realizado en alguna ocasión una compra por Internet con su tarjeta de crédito, la comunicación entre su navegador y el servidor para dicha compra tuvo lugar, casi con total seguridad, sobre SSL. (Puede verificar que su navegador está usando SSL viendo si el URL comienza por https: en lugar de por http:.)

Para entender la necesidad de SSL, analicemos un escenario de comercio electrónico típico por Internet. Imagine que Benito está navegando por la Web y que accede al sitio web de la empresa Alicia Incorporated, que se dedica a la venta de perfumes. El sitio Alicia

Incorporated muestra un formulario en el que se supone que Benito debe indicar el tipo de perfume y la cantidad deseada, junto con su dirección y el número de su tarjeta de crédito. Benito introduce esta información, hace clic en el botón Enviar y espera recibir (por correo postal ordinario) los perfumes adquiridos; también espera recibir un cargo correspondiente a ese pedido en su siguiente extracto de la tarjeta de crédito. Todo esto suena muy bien, pero si no se adopta ninguna medida de seguridad Benito podría encontrarse con algunas sor-presas.

- Si no se utiliza ningún tipo de confidencialidad (cifrado), un intruso podría interceptar el pedido de Benito y obtener la información de su tarjeta de crédito. El intruso podría entonces realizar compras a expensas de Benito.
- Si no se utiliza ningún mecanismo que garantice la integridad de los datos, un intruso podría modificar el pedido de Benito, haciéndole comprar diez veces más frascos de perfume de los deseados.
- Por último, si no se utiliza ningún tipo de autenticación del servidor, cualquier servidor podría mostrar el famoso logotipo de Alicia Incorporated cuando en realidad nos encontráramos en un sitio mantenido por Tomás, que está haciéndose pasar por Alicia Incorporated. Después de recibir el pedido de Benito, Tomás podría tomar el dinero de Benito y salir corriendo. O Tomás podría efectuar un robo de identidad, recopilando los datos relativos al nombre, la dirección y el número de tarjeta de crédito de Benito.

SSL solventa estos problemas mejorando TCP con servicios de confidencialidad, integridad de los datos, autenticación del servidor y autenticación del cliente.

SSL se emplea a menudo para proporcionar seguridad a las transacciones que tienen lugar a través de HTTP. Sin embargo, puesto que SSL dota de seguridad a TCP, puede ser empleado por cualquier aplicación que se ejecute sobre TCP. SSL proporciona una Interfaz de programación de aplicaciones (API, *Application Programmer Interface*) simple con sockets, que es similar y análoga a la API de TCP. Cuando una aplicación desea utilizar SSL, la aplicación incluye clases/bibliotecas SSL. Como se muestra en la Figura 8.25, aunque SSL reside técnicamente en la capa de aplicación, desde la perspectiva del desarrollador se trata de un protocolo de transporte que proporciona servicios TCP mejorados con servicios de seguridad.

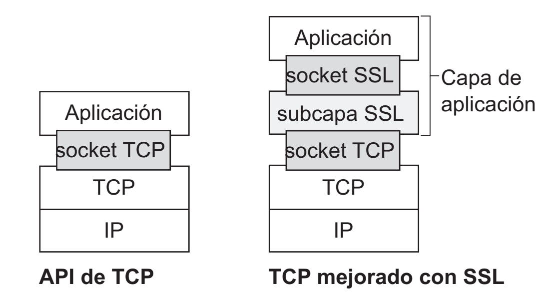


Figura 8.25 • Aunque técnicamente SSL reside en la capa de aplicación, desde la perspectiva del desarrollador se trata de un protocolo de la capa de transporte.

8.5.1 Panorámica general

Comenzaremos describiendo una versión simplificada de SSL, una que nos va a permitir obtener una panorámica general del *por qué* y el *cómo* de SSL. Haremos referencia a esta versión simplificada de SSL como "casi-SSL". Después de describir esta versión, en la siguiente subsección pasaremos a describir en detalle el protocolo SSL real. Tanto la versión casi-SSL como SSL se componen de tres fases: *acuerdo*, *deducción de la clave* y *transferencia de datos*. A continuación vamos a describir las tres fases de una sesión de comunicación entre un cliente (Benito) y un servidor (Alicia), teniendo Alicia una pareja de claves privada/pública y un certificado que asocia su identidad con su clave pública.

Fase de acuerdo

Durante la fase de acuerdo, Benito tiene que (a) establecer una conexión TCP con Alicia, (b) verificar que Alicia es *realmente* Alicia y (c) enviar a Alicia una clave secreta maestra, que ambos emplearán para generar todas las claves simétricas que necesiten para la sesión SSL. Estos tres pasos se muestran en la Figura 8.26. Observe que, una vez que se ha establecido la conexión TCP, Benito envía a Alicia un mensaje de saludo. A continuación, Alicia responde con su certificado, que contiene su clave pública. Como se ha explicado en la Sección 8.3, dado que el certificado ha sido emitido por una autoridad de certificación, Benito puede estar seguro de que la clave pública del certificado pertenece a Alicia. Benito genera entonces una clave maestra (MS, *Master Secret*), la cual sólo utilizará para esta sesión SSL; cifra la MS con la clave pública de Alicia para crear la clave maestra cifrada (EMS, *Encyrpted Master Secret*) y se la envía a Alicia. Ésta descifra la clave maestra cifrada con su clave privada para obtener la clave maestra (MS). Después de esta fase, tanto Benito como Alicia (y nadie más) conocen la clave maestra para esta sesión SSL.

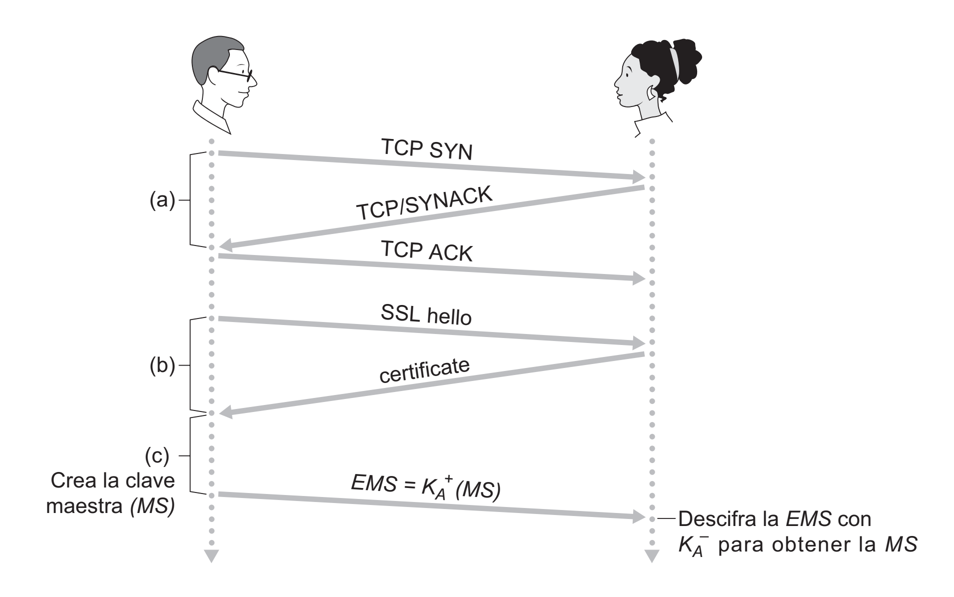


Figura 8.26 • La fase de acuerdo de casi-SSL comienza con una conexión TCP.

Deducción de las claves

En principio, la MS, ahora compartida por Benito y Alicia, podría emplearse como la clave de sesión simétrica para todos los cifrados y comprobaciones de integridad de los datos subsiguientes. Sin embargo, generalmente, se considera más seguro que Alicia y Benito utilicen claves criptográficas distintas y también que empleen claves distintas para el cifrado y las comprobaciones de integridad. Por tanto, tanto Alicia como Benito utilizan la clave maestra para generar cuatro claves:

- E_B = clave de cifrado de sesión para los datos que Benito envía a Alicia.
- M_B = clave MAC de sesión para los datos que Benito envía a Alicia.
- E_A = clave de cifrado de sesión para los datos que Alicia envía a Benito.
- M_A = clave MAC de sesión para los datos que Alicia envía a Benito.

Tanto Alicia como Benito generan las cuatro claves a partir de la clave maestra (MS). Esto podría hacerse simplemente dividiendo la clave maestra en cuatro claves (pero, como veremos, esto es un poco más complicado en el protocolo SSL *real*). Al terminar la fase de deducción de claves, tanto Alicia como Benito disponen de las cuatro claves. Las dos claves de cifrado se utilizarán para cifrar los datos y las dos claves MAC se emplearán para verificar la integridad de los datos.

Transferencia de datos

Ahora que Alicia y Benito comparten las cuatro mismas claves de sesión (E_B, M_B, E_A y M_A) pueden comenzar a enviarse datos de forma segura a través de la conexión TCP. Puesto que TCP es un protocolo de flujos de bytes, una técnica natural sería que SSL cifrara los datos de aplicación sobre la marcha y luego pasara esos datos cifrados también sobre la marcha a TCP. Pero, si hacemos esto así, ¿dónde incluiríamos el valor MAC necesario para comprobar la integridad? Realmente, no es deseable tener que esperar a que termine la sesión TCP para verificar la integridad de todos los datos que Benito ha estado enviando a lo largo de la sesión completa. Para resolver este problema, SSL divide el flujo de datos en *registros*, añade un código MAC a cada registro para comprobar la integridad y luego cifra el registro junto con el código MAC. Para crear el valor MAC, Benito introduce los datos del registro y la clave M_B en una función hash, como hemos visto en la Sección 8.3. Para cifrar el paquete formado por el registro y el valor MAC, Benito utiliza su clave de cifrado de sesión E_B. Este paquete cifrado se pasa entonces a TCP para transportarlo a través de Internet.

Aunque este método permite resolver bastantes de los problemas, sigue sin ser perfecto en lo que se refiere a proporcionar integridad de los datos para todo el flujo de mensajes. En particular, suponga que Tomás es un intruso que lleva a cabo un ataque por interposición y que tiene la capacidad de insertar, borrar y sustituir segmentos en el flujo de segmentos TCP enviados entre Alicia y Benito. Tomás, por ejemplo, podría capturar dos segmentos enviados por Benito, invertir el orden de los mismos, ajustar los números de secuencia TCP (que no están cifrados) y luego enviar los dos segmentos en orden inverso a Alicia. Suponiendo que cada segmento TCP encapsula exactamente un registro, vamos a ver ahora cómo procesaría Alicia dichos segmentos.

1. El TCP que se ejecuta en Alicia pensará que todo es correcto y pasará los dos registros a la subcapa SSL.

2. SSL en Alicia descifrará los dos registros.
3. SSL en Alicia utilizaría la clave MAC en cada registro para verificar la integridad de los datos de los dos registros.
4. SSL pasaría a continuación los flujos de bytes descifrados de los dos registros a la capa de aplicación; pero el flujo de bytes completo recibido por Alicia no estaría en el orden correcto debido a la inversión del orden de los registros.

Animamos al lector a examinar escenarios similares para los casos en que Tomás elimine o reproduzca segmentos.

La solución a este problema, como probablemente ya habrá imaginado, consiste en utilizar números de secuencia. SSL hace esto de la forma siguiente: Benito mantiene un contador de número de secuencia, que inicializa en cero y que incrementa cada vez que envía un registro SSL. Benito no incluye realmente un número de secuencia en el propio registro, sino que cuando calcula el código MAC incluye el número secuencia en el cálculo del código MAC. Así, ahora el valor MAC es un hash de los datos más la clave MAC M_B *más el número de secuencia actual.* Alicia controla los números de secuencia de Benito, pudiendo verificar la integridad de los datos de un registro incluyendo el número de secuencia apropiado en el cálculo de MAC. Este uso de los números de secuencia SSL impide que Tomás lleve a cabo un ataque por interposición, tal como la reordenación o reproducción de segmentos (¿por qué?).

Registro SSL

En la Figura 8.27 se muestra el registro SSL (así como el registro casi-SSL). El registro consta de un campo de tipo, un campo de versión, un campo de longitud, un campo de datos y un campo MAC. Observe que los tres primeros campos no están cifrados. El campo de tipo indica si el registro es un mensaje de la fase de acuerdo o un mensaje que contiene datos de aplicación. También se utiliza para cerrar la conexión SSL, como explicamos más adelante. SSL en el terminal receptor utiliza el campo de longitud para extraer los registros SSL del flujo de bytes TCP entrante. El campo de versión se explica por sí mismo.

8.5.2 Una panorámica más completa

En la subsección anterior nos hemos ocupado del protocolo casi-SSL; esto ha servido para proporcionarnos un conocimiento básico acerca del por qué y del cómo de SSL. Ahora que ya tenemos una idea básica de SSL, podemos profundizar un poco y examinar los fundamentos del protocolo SSL real. En paralelo con la lectura de esta descripción del protocolo SSL, le animamos a que complete la práctica de laboratorio acerca de SSL con Wireshark, disponible en el sitio web del libro.

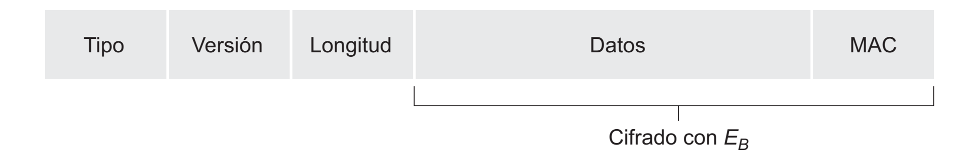


Figura 8.27 • Formato de registro para SSL.

Fase de acuerdo de SSL

SSL no obliga a que Alicia y Benito utilicen un algoritmo de clave simétrica específico, un algoritmo de clave pública específico ni un código MAC específico. En su lugar, SSL permite que Alicia y Benito acuerden al principio de la sesión SSL, durante la fase de acuerdo, los algoritmos criptográficos que van a emplear. Además, durante la fase de acuerdo, Alicia y Benito se intercambian números distintivos, que se utilizan en la creación de las claves de sesión (E_B, M_B, E_A y M_A). Los pasos de la fase de acuerdo del protocolo SSL real son los siguientes:

1. El cliente envía la lista de algoritmos criptográficos que soporta, junto con un número distintivo de cliente.
2. A partir de la lista, el servidor elige un algoritmo simétrico (por ejemplo, AES), un algoritmo de clave pública (por ejemplo, RSA con una longitud específica de clave) y un algoritmo MAC. Devuelve al cliente las elecciones que ha hecho, así como un certificado y un número distintivo de servidor.
3. El cliente verifica el certificado, extrae la clave pública del servidor, genera una clave pre-maestra (PMS, *Pre-Master Secret*), cifra la PMS con la clave pública del servidor y envía la PMS cifrada al servidor.
4. Utilizando la misma función de deducción de clave (como la especificada por el estándar SSL), el cliente y el servidor calculan independientemente la clave maestra (MS) a partir de la PMS y de los números distintivos. La PMS se divide entonces para generar las dos claves de cifrado y las dos claves MAC. Además, cuando el cifrado simétrico elegido emplea CBC (tal como 3DES o AES) también se obtienen a partir de la PMS dos vectores de inicialización (IV), uno para cada lado de la conexión. A partir de este momento todos los mensajes intercambiados entre el cliente y el servidor son cifrados y autenticados (con un código MAC).
5. El cliente envía un código MAC de todos los mensajes de acuerdo.
6. El servidor envía un código MAC de todos los mensajes de acuerdo.

Los dos últimos pasos protegen el procedimiento de acuerdo frente a posibles modificaciones de los datos. Para comprobarlo, observe que en el paso 1 el cliente normalmente ofrece una lista de algoritmos (algunos de ellos fuertes y otros más débiles). Esta lista de algoritmos se envía como texto en claro, dado que todavía no se han acordado los algoritmos de cifrado y las claves. Tomás, como atacante interpuesto (*man-in-the-middle*), podría borrar los algoritmos más fuertes de la lista obligando al cliente a seleccionar un algoritmo débil. Para evitar un ataque por modificación, en el paso 5 el cliente envía un valor MAC de la concatenación de todos los mensajes de acuerdo que ha enviado y ha recibido. El servidor puede comparar dicho valor MAC con el valor MAC de los mensajes de acuerdo que haya recibido y enviado. Si existe alguna incoherencia, el servidor puede terminar la conexión. De forma similar, el servidor envía un mensaje MAC de los mensajes de acuerdo que ha visto, permitiendo al cliente detectar cualquier incoherencia.

El lector puede estar preguntándose por qué se introducen valores distintivos en los pasos 1 y 2. ¿No bastaría con utilizar números de secuencia para impedir los ataques por reproducción de segmentos? La respuesta es que sí, pero esos números de secuencia no impiden, por sí mismos, los "ataques por reproducción de la conexión". Considere, por ejemplo, el siguiente ataque por reproducción de la conexión: suponga que Tomás captura

todos los mensajes intercambiados entre Alicia y Benito. Al día siguiente, Tomás se hace pasar por Benito y envía a Alicia exactamente la misma secuencia de mensajes que Benito le envió el día anterior. Si Alicia no utiliza números distintivos responderá con exactamente la misma secuencia de mensajes que envió el día anterior. Alicia no sospechará nada raro, ya que cada mensaje que reciba pasará las comprobaciones de integridad. Si Alicia es un servidor de comercio electrónico, pensará que Benito está realizando un segundo pedido (solicitando exactamente los mismos artículos). Por otro lado, incluyendo un número distintivo en el protocolo Alicia enviará números distintivos diferentes en cada sesión TCP, haciendo que las claves de cifrado sean distintas en cada uno de los dos días. Por tanto, cuando Alicia reciba una serie de registros SSL reproducidos procedentes de Tomás, esos registros no pasarán las comprobaciones de integridad y la transacción de comercio electrónico falsa no llegará a completarse. En resumen, en SSL los números distintivos se emplean para defenderse de los "ataques por reproducción de la conexión", mientras que los números de secuencia se emplean para defenderse frente a la reproducción de paquetes individuales durante un sesión activa.

Cierre de la conexión

En algún momento, Benito o Alicia querrán terminar la sesión SSL. Una posible técnica consistiría en dejar que Benito terminara la sesión SSL simplemente terminando la conexión TCP subyacente; es decir, hacer que Benito envíe un segmento TCP FIN a Alicia. Pero ese diseño tan simplista abre la puerta a los *ataques de truncamiento* en los que Tomás se introduce de nuevo en mitad de una sesión SSL activa y termina la sesión prematuramente con un segmento TCP FIN. Si Tomás hiciera esto, Alicia pensaría que ha recibido todos los datos de Benito, cuando en realidad sólo ha recibido una parte de los mismos. La solución a este problema consiste en indicar en el campo de tipo si el registro sirve para terminar la sesión SSL. (Aunque el tipo SSL se envía como texto en claro, siempre es autenticado en el receptor utilizando el valor MAC del registro.) Incluyendo dicho campo, si Alicia recibiera un segmento TCP FIN antes de recibir un registro SSL de cierre deduciría inmediatamente que algo raro está sucediendo.

Esto completa nuestra introducción a SSL. Hemos visto que esta tecnología utiliza muchos de los principios criptográficos explicados en la Secciones 8.2 y 8.3. Los lectores que deseen explorar SSL a un nivel más profundo pueden consultar el libro de Rescorla sobre SSL, que es bastante cómodo de leer [Rescorla 2001].

8.6 Seguridad de la capa de red: IPsec y redes privadas virtuales

El protocolo de seguridad IP, más conocido como **IPsec**, proporciona seguridad en la capa de red. IPsec proporciona seguridad a los datagramas IP intercambiados por cualesquiera dos entidades de la capa de red, incluyendo hosts y routers. Como enseguida veremos, muchas instituciones (corporaciones, agencias gubernamentales, organizaciones sin ánimo de lucro, etc.) utilizan IPsec para crear **redes privadas virtuales (VPN)**, que funcionan sobre la red Internet pública.

Antes de entrar en los detalles específicos de IPsec, demos un paso atrás y consideremos qué es lo que implica proporcionar confidencialidad en la capa de red. Con la confidenciali-

dad en la capa de red entre una pareja de entidades de red (por ejemplo, entre dos routers, entre dos hosts o entre un router y un host) la entidad emisora cifra las cargas útiles de todos los datagramas que envíe hacia la entidad receptora. La carga útil cifrada podría ser un segmento TCP, un segmento UDP, un mensaje ICMP, etc. Si dispusiéramos de tal servicio de la capa de red, todos los datos enviados de una entidad a la otra (incluyendo los mensajes de correo electrónico, las páginas web, los mensajes de acuerdo TCP y los mensajes de administración, como ICMP y SNMP) estarían ocultos a ojos de posibles terceros que pudieran estar husmeando los mensajes que circulan por la red. Por esta razón, decimos que la seguridad de la capa de red proporciona un servicio básico de "ocultación".

Además de la confidencialidad, un protocolo de seguridad de la capa de red podría potencialmente proporcionar otros servicios de seguridad. Por ejemplo, podría ofrecer mecanismos de autenticación del origen de modo que la entidad receptora pueda verificar cuál es el origen del datagrama seguro. Un protocolo de seguridad de la capa de red podría proporcionar un servicio de integridad de los datos de modo que la entidad receptora pueda comprobar si se ha producido alguna alteración del datagrama mientras éste se encontraba en tránsito. Un servicio de seguridad de la capa de red también podría proporcionar mecanismos para prevenir ataques por reproducción, lo que significa que Benito podría detectar cualquier datagrama duplicado que un atacante pudiera insertar. Como pronto veremos, IPsec de hecho proporciona mecanismos para todos estos servicios de seguridad, es decir, para la confidencialidad, la autenticación de origen, la integridad de los datos y la prevención de los ataques por reproducción.

8.6.1 IPsec y redes privadas virtuales (VPN)

Normalmente, una institución que abarque múltiples regiones geográficas deseará disponer de su propia red IP, de modo que sus hosts y servidores puedan intercambiarse datos de forma segura y confidencial. Para conseguir este objetivo, esta institución podría implantar realmente una red física independiente (incluyendo routers, enlaces y una infraestructura DNS) que esté completamente separada de la red Internet pública. Dicha red separada, dedicada a una institución concreta, se denomina **red privada**. No es sorprendente que tales redes privadas puedan llegar a ser muy costosas, ya que la institución necesitará comprar, instalar y mantener su propia infraestructura física de red.

En lugar de implantar y mantener una red privada, muchas instituciones crean actualmente redes VPN sobre la red Internet pública existente. Con una VPN el tráfico entre sucursales se envía a través de la red Internet pública, en lugar de enviarse a través de una red físicamente independiente. Pero para proporcionar confidencialidad, el tráfico entre sucursales se cifra antes de entrar en la Internet pública. En la Figura 8.28 se muestra un ejemplo simple de red VPN. Aquí, la institución está compuesta por una oficina principal, una sucursal y una serie de vendedores itinerantes que suelen acceder a Internet desde la habitación de su hotel. (En la figura sólo se muestra uno de esos vendedores.) En esta VPN, cuando dos hosts situados en la oficina principal se intercambian datagramas IP o cuando dos hosts de la sucursal quieren comunicarse utilizan el protocolo simple y tradicional IPv4 (es decir, sin servicios IPsec). Sin embargo, cuando dos hosts de la institución se comunican a través de una ruta que atraviesa la red Internet pública, el tráfico se cifra antes de entrar en Internet.

Para entender cómo funciona una red VPN, veamos un ejemplo simple en el contexto de la Figura 8.28. Cuando un host de la oficina principal envía un datagrama IP a un vendedor que se encuentra en un hotel, el router de pasarela de la oficina principal convierte el datagrama IPv4 simple en un datagrama IPsec y luego reenvía dicho datagrama IPsec hacia Inter-

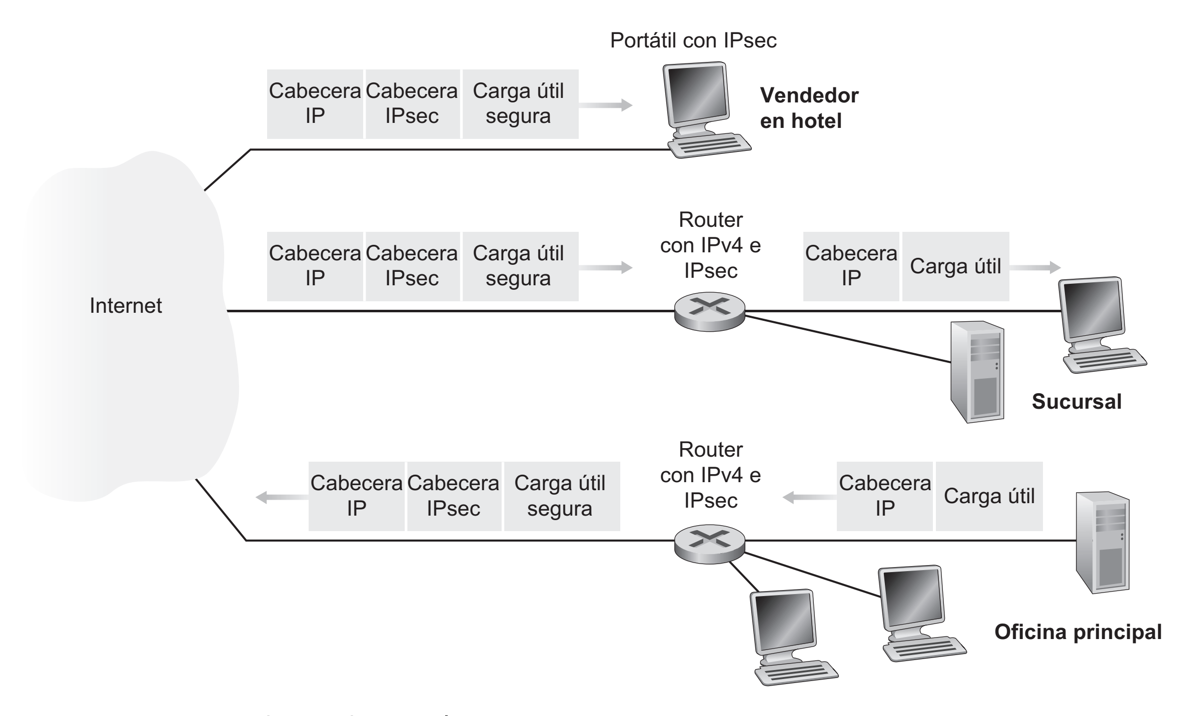


Figura 8.28 • Red privada virtual (VPN).

net. Este datagrama IPsec tiene de hecho una cabecera IPv4 tradicional, de modo que los routers de la red Internet pública procesan el datagrama como si se tratara de un datagrama IPv4 normal; para ellos el datagrama es, de hecho, como cualquier otro. Pero como se muestra en la Figura 8.28, la carga útil del datagrama IPsec incluye una cabecera IPsec, que es utilizada para el procesamiento IPsec; además, la carga útil del datagrama IPsec está cifrada. Cuando el datagrama IPsec llega al portátil del vendedor, el sistema operativo del equipo descifra la carga útil y proporciona algunos otros servicios de seguridad, como la verificación de la integridad de los datos, y pasa la carga útil descifrada hacia el protocolo de la capa superior (por ejemplo, hacia TCP o UDP).

Esto es sólo una pequeña panorámica de cómo una institución podría utilizar IPsec para crear una red VPN. Para que los árboles no nos oculten el bosque, hemos dejado conscientemente de lado muchos detalles importantes. Realicemos ahora un examen más detallado.

8.6.2 Los protocolos AH y ESP

IPsec es un protocolo bastante complejo que está definido en más de una docena de documentos RFC. Dos documentos importantes son RFC 4301, que describe la arquitectura global de seguridad IP, y RFC 2411, que proporciona una panorámica de la serie de protocolos IPsec. Como siempre, nuestro objetivo en este libro de texto no es simplemente repetir los arcanos y áridos documentos RFC, sino más bien adoptar un enfoque más operativo y pedagógico a la hora de describir los protocolos.

En la serie de protocolos IPsec hay dos protocolos principales: el protocolo de **Cabecera de autenticación (AH, *Authentication Header*)** y el protocolo de **Carga útil de seguridad para encapsulación (ESP, *Encapsulation Security Payload*)**. Cuando una entidad

IPsec de origen (normalmente un host o un router) envía datagramas seguros a una entidad de destino (también un host o un router) lo hace con el protocolo AH o el protocolo ESP. El protocolo AH proporciona autenticación del origen e integridad de los datos, pero *no* proporciona confidencialidad. El protocolo ESP proporciona autenticación del origen, integridad de los datos *y* confidencialidad. Puesto que la confidencialidad a menudo es crítica para las redes VPN y otras aplicaciones IPsec, el protocolo ESP se utiliza mucho más ampliamente que el protocolo AH. Con el fin de desmitificar IPsec y evitar buena parte de las complicaciones asociadas nos vamos por tanto a centrar exclusivamente en el protocolo ESP. Aquellos lectores que deseen también aprender los fundamentos del protocolo AH pueden explorar los documentos RFC y otros recursos en línea.

8.6.3 Asociaciones de seguridad

Los datagramas IPsec se intercambian entre parejas de entidades de red, como por ejemplo entre dos hosts, entre dos routers o entre un host y un router. Antes de enviar datagramas IPsec desde la entidad de origen a la de destino, ambas entidades crean una conexión lógica en la capa de red. Esta conexión lógica se denomina **asociación de seguridad (SA, *Security Association*)**. Una asociación de seguridad es una conexión lógica de tipo simplex; es decir, una conexión unidireccional desde el origen al destino. Si ambas entidades desean enviarse datagramas seguros entre sí, entonces será necesario establecer dos SA (es decir, dos conexiones lógicas), una en cada dirección.

Por ejemplo, considere de nuevo la VPN institucional de la Figura 8.28. Esta institución consta de una oficina principal, una sucursal y un cierto número, por ejemplo, n, de vendedores itinerantes. Supongamos, como ejemplo, que existe tráfico IPsec bidireccional entre la oficina principal y la sucursal y entre la oficina principal y los vendedores. En esta VPN, ¿cuántas asociaciones de seguridad existirían? Para responder a esta cuestión, observe que hay dos SA entre el router de pasarela de la oficina principal y el router de pasarela de la sucursal (una en cada dirección); para la computadora portátil de cada vendedor también habrá dos SA entre el router de pasarela de la oficina principal y el portátil (de nuevo, una en cada dirección). Por tanto, en total, habrá $(2 + 2n)$ asociaciones de seguridad. *Sin embargo, recuerde que no todo el tráfico enviado hacia Internet por los routers de pasarela o por las computadoras portátiles estará protegido mediante IPsec.* Por ejemplo, un host situado en la oficina principal podría querer acceder a un servidor web (como Amazon o Google) disponible en la red Internet pública. Por tanto, el router de pasarela (y los portátiles) enviará hacia Internet tanto datagramas IPv4 normales como datagramas dotados de seguridad IPsec.

Tratemos ahora de examinar las interioridades de una asociación de seguridad. Para que las explicaciones sean tangibles y concretas vamos hacerlo en el contexto de una asociación de seguridad existente entre el router R1 y el router R2 de la Figura 8.29. (Podemos considerar que el router R1 es el router de pasarela de la oficina principal y que el router R2 es el router de pasarela de la sucursal en el contexto de la Figura 8.28.) El router R1 mantendrá una cierta información de estado acerca de esta SA, la cual incluirá:

- Un identificador de 32 bits para la SA, denominado **Índice de parámetro de seguridad (SPI, *Security Parameter Index*)**.
- La interfaz de origen de la SA (en este caso, 200.168.1.100) y la interfaz de destino de la SA (en este caso 193.68.2.23).
- El tipo de cifrado que se va a utilizar (por ejemplo, 3DES con CBC).

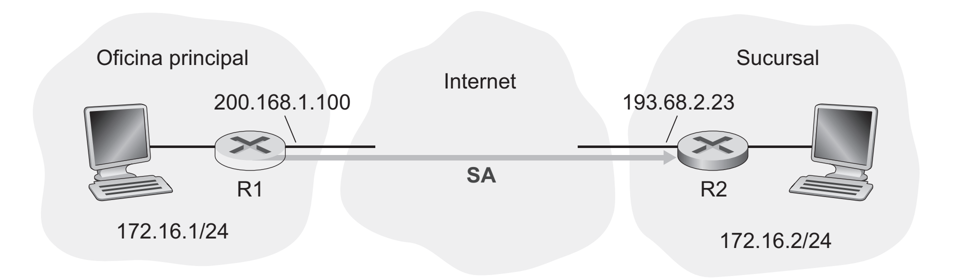


Figura 8.29 • Asociación de seguridad (SA) de R1 a R2.

- La clave de cifrado.
- El tipo de comprobación de integridad (por ejemplo, HMAC con MD5).
- La clave de autenticación.

Cada vez que el router R1 necesite construir un datagrama IPsec para reenviarlo a través de esta SA, accederá a esta información de estado para determinar cómo debe autenticar y cifrar el datagrama. De forma similar, el router R2 mantendrá la misma información de estado para esta SA y utilizará esta información para autenticar y descifrar todos los datagramas IPsec que lleguen desde dicha asociación de seguridad.

Cada entidad IPsec (router o host) suele mantener información de estado para muchas asociaciones de seguridad. Por ejemplo, en el ejemplo de la red VPN de la Figura 8.28 con n vendedores, el router de pasarela de la oficina principal mantiene información de estado para $(2 + 2n)$ asociaciones de seguridad. Cada entidad IPsec almacena la información de estado para todas sus asociaciones de seguridad en su **Base de datos de asociaciones de seguridad (SAD, *Security Association Database*)**, que es una estructura de datos contenida en el kernel del sistema operativo de esa entidad.

8.6.4 El datagrama IPsec

Habiendo descrito las asociaciones de seguridad, podemos ahora describir la estructura real del datagrama IPsec. IPsec tiene dos formas distintas de paquete, una para el denominado **modo túnel** y otra para el denominado **modo transporte**. El modo túnel, al ser más apropiado para las redes VPN, está más ampliamente implantado que el modo transporte. Con el fin de desmitificar todavía más IPsec y evitar buena parte de los aspectos más complejos, nos vamos a centrar por tanto exclusivamente en el modo túnel. Una vez que tenga una sólida comprensión de dicho modo, el lector debería poder aprender por su cuenta los detalles acerca del modo transporte.

El formato de paquete del datagrama IPsec se muestra en la Figura 8.30. Puede que el lector crea que los formatos de paquete son aburridos e insípidos, pero pronto comprobará que el datagrama IPsec tiene en realidad la apariencia y el sabor de un manjar tex-mex. Examinemos los campos IPsec en el contexto de la Figura 8.29. Suponga que el router R1 recibe un datagrama IPv4 normal procedente del host 172.16.1.17 (situado en la red de la oficina principal) que está destinado al host 172.16.2.48 (situado en la red de la sucursal). El router R1 utiliza la siguiente receta para convertir este “datagrama IPv4 original” en un datagrama IPsec:

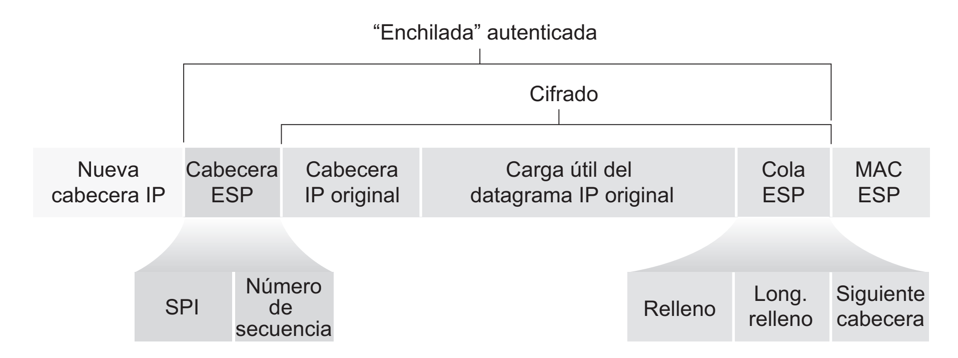


Figura 8.30 • Formato del datagrama IPsec.

- Añade al final del datagrama IPv4 original (¡que incluye los campos originales de cabecera!) un campo de "cola ESP".
- Cifra el resultado utilizando el algoritmo y la clave especificados por la asociación de seguridad.
- Añade al principio de este paquete cifrado un campo denominado "cabecera ESP"; el paquete resultante se conoce como "enchilada".
- Crea un valor MAC de autenticación para *toda la enchilada* utilizando el algoritmo y la clave especificados en la SA.
- Añade el valor MAC al final de la enchilada formando así la *carga útil.*
- Por último, crea una nueva cabecera IP con todos los campos clásicos de la cabecera IPv4 (que suman normalmente 20 bytes de longitud) y añade dicha cabecera al principio de la carga útil.

Observe que el datagrama IPsec resultante es un datagrama IPv4 perfectamente normal, con los campos tradicionales de cabecera IPv4 seguidos de una carga útil. Pero en este caso la carga útil contiene una cabecera ESP, el datagrama IP original, una cola ESP y un campo de autenticación ESP (estando cifrados el datagrama original y la cola ESP). El datagrama IP original tiene el valor 172.16.1.17 como dirección IP de origen y el 172.16.2.48 como dirección IP de destino. Puesto que el datagrama IPsec incluye el datagrama IP original, estas direcciones se incluyen (y se cifran) como parte de la carga útil del paquete IPsec. ¿Pero qué sucede con las direcciones IP de origen y de destino contenidas en la nueva cabecera IP, es decir, en la cabecera situada más a la izquierda en el datagrama IPsec? Como cabría esperar, esos valores se configuran con las direcciones de las interfaces de router de origen y de destino situadas en los dos extremos de los túneles, es decir, con los valores 200.168.1.100 y 193.68.2.23. Asimismo, el número de protocolo en este nuevo campo de cabecera IPv4 no se configura con el valor correspondiente a TCP, UDP o SMTP, sino con el valor 50, que indica que se trata de un datagrama IPsec que está empleando el protocolo ESP.

Después de que R1 envíe el datagrama IPsec hacia la red Internet pública, pasará a través de muchos routers antes de alcanzar R2. Cada uno de estos routers procesará el datagrama como si fuera un datagrama normal; de hecho, todos esos routers no son conscientes de que el datagrama esté transportando datos cifrados mediante IPsec. Para estos routers de

la red Internet pública, puesto que la dirección IP de destino contenida en la cabecera externa es R2, el destino último del datagrama es R2.

Habiendo examinado este ejemplo de cómo se construye un datagrama IPsec, veamos ahora con más detalle los ingredientes de la enchilada. Como podemos ver en la Figura 8.30, la cola ESP está compuesta por tres campos: relleno, longitud de relleno y siguiente cabecera. Recuerde que los sistemas de cifrado de bloque requieren que el mensaje que hay que cifrar sea un múltiplo entero de la longitud de bloque. Por ello se emplea un relleno (compuesto por bytes que no tienen ningún significado) para que, al añadirlo al datagrama original (junto con los campos de longitud de relleno y de siguiente cabecera), el "mensaje" resultante tenga un número entero de bloques. El campo de longitud de relleno indica a la entidad receptora cuánto relleno se ha insertado (y, por tanto, cuánto relleno habrá que eliminar). El campo de siguiente cabecera indica el tipo (por ejemplo, UDP) de los datos contenidos en el campo de datos de carga útil. Los datos de carga útil (normalmente, el datagrama IP original) y la cola ESP se concatenan y se cifran.

Delante de esta unidad cifrada se encuentra la cabecera ESP, que se envía como texto en claro y que consta de dos campos: el SPI y el campo de número de secuencia. El SPI indica a la entidad receptora cuál es la SA a la que pertenece el datagrama; la entidad receptora puede entonces indexar su base de datos SAD con el índice SPI para determinar los algoritmos y claves apropiados de autenticación/descifrado. El campo de número de secuencia se utiliza para defenderse frente a los ataques por reproducción.

La entidad emisora también añade un código MAC de autenticación. Como hemos dicho anteriormente, la entidad emisora calcula un código MAC para toda la enchilada (compuesta por la cabecera ESP, el datagrama IP original y la cola ESP, estando el datagrama y la cola cifrados). Recuerde que para calcular un valor MAC, el emisor añade una clave secreta MAC a la enchilada y luego calcula un valor hash de longitud fija para el resultado.

Cuando R2 recibe el datagrama IPsec, observa que la dirección IP de destino del datagrama es el propio R2, por lo que dicho router se encarga de procesar el datagrama. Puesto que el campo de protocolo (en la cabecera IP situada más a la izquierda) tiene el valor 50, R2 ve que debe aplicar el procesamiento ESP de IPsec al datagrama. En primer lugar, analizando la enchilada, R2 utiliza el SPI para determinar a qué asociación de seguridad (SA) pertenece el datagrama. En segundo lugar, calcula el valor MAC de la enchilada y verifica que es coherente con el valor contenido en el campo ESP MAC. Si lo es, el router sabrá que la enchilada procede del router R1 y que no ha sido manipulada. En tercer lugar, comprueba el campo de número de secuencia para verificar que el datagrama sea reciente y no un datagrama reproducido. En cuarto lugar, descifra la unidad cifrada utilizando la clave y el algoritmo de descifrado asociados con la SA. En quinto lugar, elimina el relleno y extrae el datagrama IP normal original. Y, finalmente, en sexto lugar, reenvía el datagrama original a la red de la sucursal para que el datagrama llegue a su verdadero destino. Es una receta un tanto complicada, ¿verdad? ¡Bueno, nunca dijimos que preparar una enchilada fuera fácil!

Existe todavía otra sutileza importante que necesitamos explicar y que está centrada en la siguiente cuestión: cuando el router R1 recibe un datagrama (no dotado de seguridad) procedente de un host de la red de la oficina principal y dicho datagrama está destinado a alguna dirección IP de destino situada fuera de la oficina principal, ¿cómo sabe R1 si ese datagrama debe ser convertido en un datagrama IPsec? Y si tiene que ser procesado por IPsec, ¿cómo sabe R1 qué SA (de las muchas asociaciones de seguridad existentes en su base de datos SAD) hay que utilizar para construir el datagrama IPsec? El problema se resuelve de la

forma siguiente. Junto con una base de datos SAD, la entidad IPsec también mantiene otra estructura de datos denominada **Base de datos de políticas de seguridad (SPD, *Security Policy Database*)**. La SPD indica qué tipos de datagramas (en función de la dirección IP de origen, la dirección IP de destino y el tipo de protocolo) hay que procesar mediante IPsec; y para aquellos que haya que procesar mediante IPsec, qué SA debe emplearse. En un cierto sentido, la información de una SPD indica "qué" hacer con los datagramas que lleguen, mientras que la información de la SAD indica "cómo" hay que hacerlo.

Resumen de los servicios IPsec

¿Qué servicios proporciona IPsec exactamente? Examinemos estos servicios desde la perspectiva de un atacante, como por ejemplo Tomás, que se ha interpuesto (*man-in-the-middle*) en la comunicación, situándose en algún lugar de la ruta entre los routers R1 y R2 de la Figura 8.29. Vamos a suponer a lo largo de estas explicaciones que Tomás no conoce las claves de cifrado y de autenticación empleadas por la SA. ¿Qué cosas puede hacer Tomás y cuáles no? En primer lugar, Tomás no puede ver el datagrama original. De hecho, no sólo están los datos del datagrama original ocultos a ojos de Tomás, sino que también lo están el número de protocolo, la dirección IP de origen y la dirección IP de destino. Para los datagramas enviados a través de la SA, Tomás sólo sabe que el datagrama tiene su origen en algún host de la red 172.16.1.0/24 y que está destinado a algún host de la red 172.16.2.0/24. No sabe si está transportando datos TCP, UDP o ICMP; no sabe si está transportando HTTP, SMTP, o algún otro tipo de datos de aplicación. Esta confidencialidad, por tanto, va bastante más allá que en SSL. En segundo lugar, suponga que Tomás trata de alterar un datagrama en la SA modificando algunos de sus bits. Cuando este datagrama alterado llegue a R2 no pasará las comprobaciones de integridad (utilizando el valor MAC), desbaratando una vez más las intenciones de Tomás. En tercer lugar, suponga que Tomás intenta hacerse pasar por R1, creando un datagrama IPsec cuyo origen sea 200.168.1.100 y cuyo destino sea 193.68.2.23. El ataque de Tomás no tendrá ningún efecto, ya que este datagrama de nuevo no pasará la comprobación de integridad realizada en R2. Finalmente, puesto que IPsec incluye números de secuencia, Tomás no podrá desarrollar con éxito ningún ataque por reproducción. En resumen, y tal como dijimos al principio de esta sección, IPsec proporciona (entre cualquier pareja de dispositivos que procesen paquetes en la capa de red) mecanismos de confidencialidad, de autenticación del origen, de integridad de los datos y de prevención de los ataques por reproducción.

8.6.5 IKE: gestión de claves en IPsec

Cuando una red VPN tiene un pequeño número de puntos terminales (por ejemplo, sólo dos routers, como en la Figura 8.29), el administrador de la red puede introducir manualmente la información de la SA (claves y algoritmos de cifrado/autenticación y los índices SPI) en las bases de datos SAD en los puntos terminales. Este tipo de "introducción manual" de las claves resulta obviamente poco práctico para una VPN de gran tamaño, que puede constar de centenares o incluso miles de hosts y routers IPsec. Las tareas de implantación de gran envergadura y geográficamente distribuidas requieren un mecanismo automatizado para la creación de las SA. IPsec lleva a cabo este tipo de tarea mediante el protocolo de Intercambio de claves de Internet (IKE, *Internet Key Exchange*), especificado en RFC 4306.

IKE presenta algunas similitudes con el procedimiento de acuerdo en SSL (véase la Sección 8.5). Cada entidad IPsec tiene un certificado, que incluye la clave pública de la entidad.

Al igual que en SSL, el protocolo IKE exige que las dos entidades intercambien certificados, negocien los algoritmos de autenticación y cifrado e intercambien de modo seguro el material necesario para crear las claves de sesión para las SA de IPsec. A diferencia de SSL, IKE emplea dos fases para llevar a cabo estas tareas.

Vamos a analizar estas dos fases en el contexto de los routers R1 y R2 de la Figura 8.29. La primera fase está compuesta por dos intercambios de parejas de mensajes entre R1 y R2:

- Durante el primer intercambio de mensajes los dos lados utilizan Diffie-Hellman (véanse los problemas incluidos al final del capítulo) para crear una **IKE SA** bidireccional entre los routers. Para confundir a los estudiosos, esta SA IKE bidireccional es enteramente distinta de las SA IPsec presentadas en las Secciones 8.6.3 y 8.6.4. La IKE SA proporciona un canal autenticado y cifrado entre los dos routers. Durante este primer intercambio de parejas de mensajes se establecen las claves de cifrado y autenticación para la IKE SA. También se establece un valor secreto maestro que se utilizará para calcular las claves IPSec SA posteriormente en la fase 2. Observe que durante este primer paso no se utilizan claves pública y privada RSA. En particular, ni R1 ni R2 revelan su identidad firmando un mensaje con su clave privada.
- Durante el segundo intercambio de mensajes ambos lados se revelan mutuamente su identidad, firmando sus mensajes. Sin embargo, las identidades no son reveladas a nadie que esté husmeando pasivamente el canal de comunicación, ya que los mensajes se envían a través del canal IKE SA seguro. También durante esta fase los dos lados negocian los algoritmos de cifrado y autenticación IPsec que serán empleados por las asociaciones de seguridad IPsec.

En la fase 2 de IKE los dos lados crean una SA en cada dirección. Al final de la fase 2 las claves de sesión para cifrado y autenticación habrán sido establecidas en ambos terminales para las dos SA. Los dos lados pueden emplear entonces las SA para enviar datagramas seguros, como se describe en las Secciones 8.6.3 y 8.6.4. La principal motivación para que existan dos fases en IKE es el coste computacional: puesto que la segunda fase no implica ningún tipo de criptografía de clave pública, IKE puede generar un gran número de asociaciones de seguridad entre las dos entidades IPsec con un coste de computación relativamente bajo.

8.7 Seguridad de las redes LAN inalámbricas

La seguridad es una preocupación de particular importancia en las redes inalámbricas, en las que las ondas de radio que transportan la tramas pueden propagarse bastante más allá de los límites del edificio que alberga a los hosts y a la estación base inalámbrica. En esta sección vamos a presentar una breve introducción al tema de la seguridad inalámbrica. El lector interesado en ver un tratamiento más detallado puede leer el libro de Edney y Arbaugh [Edney 2003] que es bastante legible.

El problema de la seguridad en 802.11 ha atraído una considerable atención tanto en los círculos técnicos como en los medios de comunicación. Aunque ha habido mucha discusión en realidad se ha producido bastante poco debate, ya que parece existir un consenso universal en que la especificación 802.11 original contiene diversos fallos importantes de seguridad. De hecho, cualquiera puede descargarse actualmente software de dominio público que

aprovecha los agujeros de seguridad, haciendo que los usuarios que emplean los mecanismos de seguridad simples de 802.11 sean tan vulnerables como aquellos usuarios que no emplean ninguna característica de seguridad en absoluto.

En la siguiente sección vamos a analizar los mecanismos de seguridad inicialmente estandarizados en la especificación 802.11, que se conocen colectivamente con el nombre de **Privacidad equivalente a la del cable (WEP, *Wired Equivalent Privacy*)**. Como el nombre sugiere, WEP pretendía proporcionar un nivel de seguridad similar al que podemos encontrar en las redes cableadas. Posteriormente analizaremos algunos de los agujeros de seguridad de WEP y veremos el estándar 802.11i, que es una versión bastante más segura que 802.11 y que se adoptó en el año 2004.

8.7.1 WEP (Wired Equivalent Privacy)

El protocolo WEP del estándar IEEE 802.11 [IEEE 802.11 2009] proporciona autenticación y cifrado de datos entre un host y un punto de acceso inalámbrico (es decir, una estación base) utilizando una técnica basada en una clave simétrica compartida. WEP no especifica ningún algoritmo de gestión de claves, por lo que se presupone que el host y el punto de acceso inalámbrico han acordado de alguna manera qué clave utilizar, empleando para ello algún método fuera de banda. La autenticación se lleva a cabo de la forma siguiente:

1. Un host inalámbrico solicita la autenticación por parte de un punto de acceso.
2. El punto de acceso responde a la solicitud de autenticación con un número distintivo de 128 bytes.
3. El host inalámbrico cifra el número distintivo utilizando la clave simétrica que comparte con el punto de acceso.
4. El punto de acceso descifra el número distintivo cifrado por el host.

Si el número distintivo descifrado se corresponde con el valor del número distintivo originalmente enviado al host, entonces el host quedará autenticado por el punto de acceso.

El algoritmo de cifrado de datos de WEP se ilustra en la Figura 8.31. Se supone que tanto el host como el punto de acceso conocen una clave simétrica secreta de 40 bits, K_S. Además, se añade un vector de inicialización (IV) de 24 bits a la clave de 40 bits para crear una clave de 64 bits que se usará para cifrar una única trama. El vector IV cambiará de una trama a la siguiente, por lo que cada trama será cifrada con una clave de 64 bits distinta. El

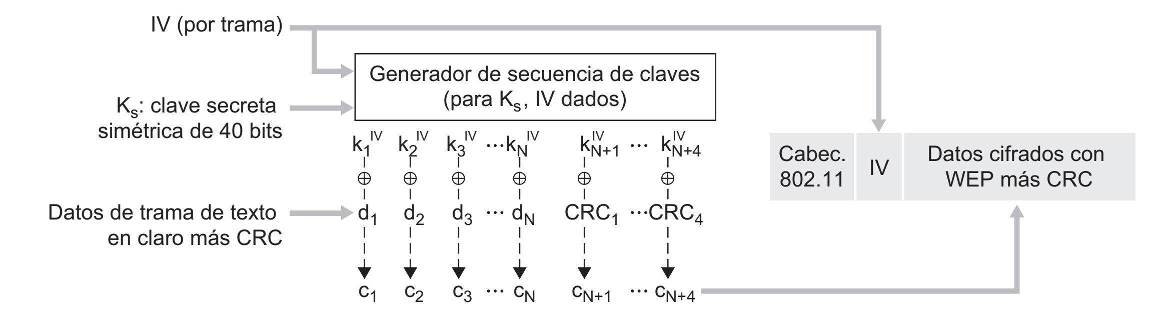


Figura 8.31 • Protocolo WEP 802.11.

cifrado se realiza de la manera siguiente. En primer lugar se calcula un valor CRC de 4 bytes (véase la Sección 5.2) para la carga útil de datos. La carga útil y los 4 bytes de CRC se cifran entonces utilizando el mecanismo de cifrado de flujo RC4. No vamos a analizar aquí los detalles de RC4 (puede encontrarlos en [Schneier 1995] y [Edney 2003]). Para nuestros propósitos, basta con saber que el algoritmo RC4, cuando se le suministra un valor de clave (en este caso, la clave (K_S, IV) de 64 bits), produce un flujo de valores de claves, k_1^{IV}, k_2^{IV}, k_3^{IV}, . . . que se utilizan para cifrar los datos y el valor de CRC de una trama. A efectos prácticos, podemos considerar que estas operaciones se realizan de byte en byte. El cifrado se lleva a cabo combinando mediante XOR el *i-ésimo* byte de datos, d_i, con el *i-ésimo* valor de clave, k_i^{IV}, perteneciente al flujo de valores de clave generado por la pareja (K_S, IV); con ello se genera el *i-ésimo* byte de texto cifrado, c_i:

$$c_i = d_i \oplus k_i^{IV}$$

El valor del vector IV cambia de una trama a la siguiente y se incluye como *texto en claro* en la cabecera de cada trama 802.11 cifrada con WEP, como se muestra en la Figura 8.31. El receptor toma la clave simétrica secreta de 40 bits que comparte con el emisor, le añade el vector IV y emplea la clave de 64 bits resultante (que es idéntica a la clave usada por el emisor para realizar el cifrado) con el fin de descifrar la trama:

$$d_i = c_i \oplus k_i^{IV}$$

El uso apropiado del algoritmo RC4 requiere que *nunca* se utilice más de una vez un mismo valor de clave de 64 bits. Recuerde que la clave WEP cambia de una trama a otra. Para un valor K_S dado (que rara vez cambia, si es que cambia alguna vez), esto quiere decir que sólo hay 2^{24} claves únicas. Si estas claves se seleccionan aleatoriamente, se puede demostrar [Walker 2000; Edney 2003] que la probabilidad de haber elegido el mismo valor del vector IV (y por tanto de haber empleado la misma clave de 64 bits) dos veces es superior al 99 por ciento después de sólo 12.000 tramas. Con un tamaño de trama de 1 kbyte y una velocidad de transmisión de 11 Mbps, bastan unos pocos segundos para transmitir 12.000 tramas. Además, puesto que el valor de IV se transmite como texto en claro dentro de la trama, un curioso que esté capturando la comunicación podrá saber si se ha utilizado un valor de IV duplicado.

Para comprender uno de los problemas que se presentan cuando se utiliza una clave duplicada, considere el siguiente ataque de texto en claro seleccionado realizado por Tomás contra Alicia. Suponga que Tomás (posiblemente utilizando alguna técnica de suplantación de dirección IP) envía una solicitud (por ejemplo, una solicitud HTTP o FTP) a Alicia para transmitir un archivo con un contenido conocido, d_1, d_2, d_3, d_4,. . . . Tomás también observa los datos cifrados c_1, c_2, c_3, c_4. . . . Puesto que $d_i = c_i \oplus k_i^{IV}$, si combinamos mediante XOR c_i con cada lado de esta igualdad obtenemos

$$d_i \oplus c_i = k_i^{IV}$$

Con esta relación, Tomás puede utilizar los valores conocidos de d_i y c_i para calcular k_i^{IV}. La siguiente vez que Tomás vea que se está utilizando el mismo valor de IV, conocerá la secuencia de clave k_1^{IV}, k_2^{IV}, k_3^{IV}, . . . y podrá, por tanto, descifrar el mensaje cifrado.

Existen también varios otros problemas adicionales de seguridad que afectan a WEP. [Fluhrer 2001] describió un ataque que aprovecha una debilidad conocida de RC4 cuando se seleccionan ciertas claves débiles. [Stubblefield 2002] analiza varias formas eficientes de

implementar y aprovechar este ataque. Otro problema relativo a WEP está relacionado con los bits CRC mostrados en la Figura 8.31 y transmitidos en la trama 802.11 para detectar posibles alteraciones de los bits de la carga útil. Sin embargo, un atacante que cambie el contenido cifrado (es decir, que sustituya los datos cifrados originales por bits sin sentido), que calcule el CRC para los bits sin sentido insertados y que coloque el CRC en una trama WEP puede generar una trama 802.11 que será aceptada por el receptor. Lo que hacen falta aquí son técnicas de integridad de mensajes como las que hemos estudiado en la Sección 8.3 para detectar la alteración o sustitución del contenido. Para conocer más detalles acerca de la seguridad en WEP, consulte [Edney 2003; Walker 2000; Weatherspoon 2000; 802.11 Security 2009] y las referencias en ellos contenidas.

8.7.2 IEEE 802.11i

Poco después de la publicación en 1999 de la norma IEEE 802.11 comenzó el trabajo para desarrollar una versión nueva y mejorada de 802.11 con mecanismos de seguridad más robustos. El nuevo estándar, conocido con el nombre de 802.11i, obtuvo la ratificación final en 2004. Como veremos, mientras que WEP proporcionaba un cifrado relativamente débil, junto con una única forma de llevar a cabo la autenticación y ningún mecanismo de distribución de claves, IEEE 802.11i proporciona formas mucho más fuertes de cifrado, un conjunto ampliable de mecanismos de autenticación y un mecanismo de distribución de claves. De aquí en adelante vamos a presentar una panorámica de 802.11i; puede encontrar una excelente presentación técnica (un flujo de audio) de 802.11i en [TechOnline 2004].

La Figura 8.32 presenta el marco conceptual de 802.11i. Además del punto de acceso y del cliente inalámbrico, 802.11i define un servidor de autenticación con el que el AP puede

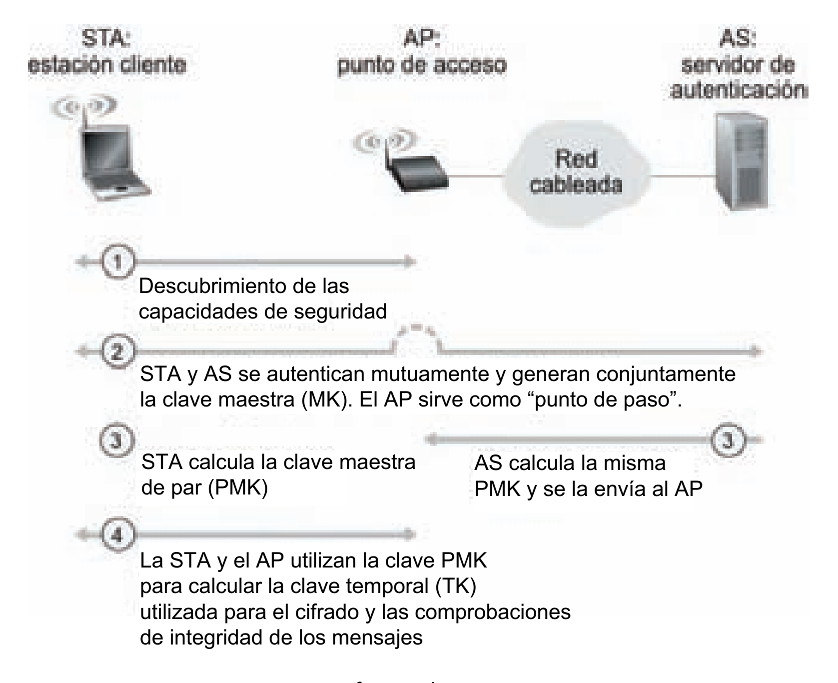


Figura 8.32 • 802.11i: cuatro fases de operación.

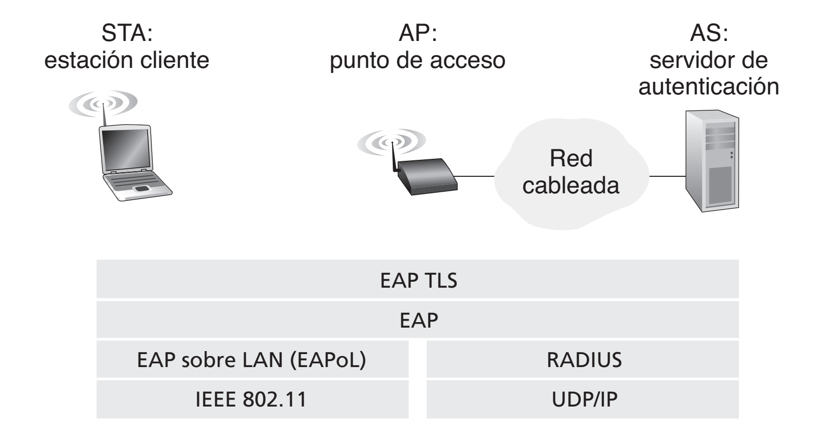


Figura 8.33 • EAP es un protocolo terminal a terminal. Los mensajes EAP se encapsulan utilizando EAPoL sobre el enlace inalámbrico existente entre el cliente y el punto de acceso, y empleando RADIUS sobre UDP/IP entre el punto de acceso y el servidor de autenticación.

comunicarse. Separar el servidor de autenticación del AP permite que un mismo servidor de autenticación preste servicio a muchos puntos de acceso, centralizando las (a menudo sensibles) decisiones concernientes a la autenticación y al acceso dentro de ese único servidor y manteniendo a un nivel bajo el coste y la complejidad de los puntos de acceso. 802.11i opera en cuatro fases:

1. *Descubrimiento.* En la fase de descubrimiento el AP anuncia su presencia y las formas de autenticación y cifrado que puede proporcionar al nodo de cliente inalámbrico. El cliente solicita entonces las formas específicas de autenticación y cifrado que desea. Aunque el cliente y el AP ya están intercambiando mensajes, el cliente no habrá sido todavía autenticado ni dispondrá aún de una clave de cifrado, por lo que son necesarios varios pasos más antes de que el cliente pueda comunicarse con un host remoto arbitrario a través del canal inalámbrico.
2. *Autenticación mutua y generación de la clave maestra (MK, Master Key).* La autenticación tiene lugar entre el cliente inalámbrico y el servidor de autenticación. En esta fase, el punto de acceso actúa básicamente como repetidor, reenviando los mensajes entre el cliente y el servidor de autenticación. El **Protocolo ampliable de autenticación (EAP, *Extensible Authentication Protocol*)** [RFC 2284] define los formatos de los mensajes terminal a terminal utilizados en un modo de interacción simple de tipo solicitud/respuesta entre el cliente y el servidor de autenticación. Como se muestra en la Figura 8.33, los mensajes EAP se encapsulan utilizando **EAPoL** (EAP sobre LAN, [IEEE 802.1X]) y se envían a través del enlace inalámbrico 802.11. Estos mensajes EAP son entonces desencapsulados en el punto de acceso y luego re-encapsulados utilizando el protocolo **RADIUS** para su transmisión sobre UDP/IP hacia el servidor de autenticación. Aunque el protocolo [RFC 2865] y el servidor RADIUS no son requeridos obligatoriamente por el protocolo 802.11i, son componentes estándar *de facto* para 802.11i.

Es posible que el protocolo **DIAMETER** [RFC 3588] recientemente estandarizado sustituya a RADIUS en un futuro próximo.

Con EAP el servidor de autenticación puede seleccionar una de varias formas para llevar a cabo la autenticación. Aunque 802.11i no impone un método de autenticación concreto, a menudo se emplea el esquema de autenticación EAP-TLS [RFC 2716]. EAP-TLS utiliza técnicas de clave pública (incluyendo cifrado con números distintivos y resúmenes de mensajes) similares a las que hemos estudiado en la Sección 8.3 para permitir que el cliente y el servidor de autenticación se autentiquen mutuamente entre sí y para calcular una clave maestra (MK) que será conocida por ambas partes.

3. *Generación de la clave maestra de par (PMK, Pairwise Master Key).* La MK es un secreto compartido que sólo conocen el cliente y el servidor de autenticación y que ambos emplean para generar una segunda clave, la clave maestra de par (PMK). El servidor de autenticación envía entonces la PMK al AP. ¡Aquí es donde queríamos llegar! El cliente y el AP ahora disponen de una clave compartida (recuerde que, en WEP, el problema de la distribución de claves ni siquiera se contemplaba) y se habrán autenticado mutuamente entre sí. Ya están casi listos para poder comenzar con la operación real.
4. *Generación de la clave temporal (TK, Temporal Key).* Con la PMK, el cliente inalámbrico y el AP pueden ahora generar claves adicionales que se utilizarán para la comunicación. De particular interés es la clave temporal (TK) que se utilizará para realizar el cifrado de nivel de enlace de los datos enviados a través del enlace inalámbrico hacia un host remoto arbitrario.

802.11i proporciona varias formas de cifrado, incluyendo el esquema de cifrado basado en AES y una versión más fuerte del cifrado WEP.

8.8 Seguridad operacional: cortafuegos y sistemas de detección de intrusiones

Hemos visto a lo largo de este capítulo que Internet no es un lugar muy seguro (los malos están ahí fuera, infligiendo toda clase de estragos). Conocida la naturaleza hostil de Internet, vamos a considerar ahora la red de una organización y al administrador de red que la gestiona. Desde el punto de vista de un administrador de red, el mundo se divide de forma bastante nítida en dos bandos: los buenos (aquellos que pertenecen a la red de la organización y que deben poder acceder a los recursos internos de la misma de una forma relativamente poco restringida) y los malos (todos los demás, aquellos que deben ser cuidadosamente escrutados a la hora de acceder a los recursos de la red). En muchas organizaciones, desde los castillos medievales a los modernos edificios de oficinas, existe un único punto de entrada/salida donde se hace una comprobación de seguridad tanto de los buenos como de los malos que entran y salen de la organización. En un castillo esto se hacía en la puerta situada en el extremo de un puente levadizo; en un edificio de oficinas, esto se hace en el control de seguridad de entrada. En una red de computadoras, cuando se comprueba si el tráfico que entra/sale de la red es seguro, cuando se registra ese tráfico y cuando se elimina o reenvía, quienes se encargan de esas tareas son una serie de dispositivos operacionales conocidos como cortafuegos, sistemas de detección de intrusiones (IDS, *Intrusion Detection System*) y sistemas de prevención de intrusiones (IPS, *Intrusion Prevention System*).

8.8.1 Cortafuegos

Un **cortafuegos** es una combinación de hardware y software que aísla la red interna de la organización de Internet, permitiendo pasar a algunos paquetes y bloqueando a otros. Un cortafuegos permite a un administrador de red controlar el acceso entre el mundo exterior y los recursos dentro de la red administrada, gestionando el flujo de tráfico hacia y desde esos recursos. Un cortafuegos tiene tres objetivos:

- *Todo el tráfico que va del exterior hacia el interior de la red, y viceversa, pasa a través del cortafuegos*. La Figura 8.34 muestra un cortafuegos, que se encuentra en el límite entre la red administrada y el resto de Internet. Aunque las organizaciones de gran tamaño pueden utilizar varios niveles de cortafuegos o cortafuegos distribuidos [Skoudis 2006], colocar un cortafuegos en un único punto de acceso a la red, como se muestra en la Figura 8.34, facilita la gestión y el imponer una política de control de acceso.
- *Sólo se permite el paso del tráfico autorizado de acuerdo con la política de seguridad local*. Con todo el tráfico de entrada y de salida de la red institucional pasando a través del cortafuegos, éste puede restringir el acceso al tráfico autorizado.
- *El propio cortafuegos es inmune a la penetración*. El propio cortafuegos es un dispositivo conectado a la red. Si no está diseñado o instalado apropiadamente puede verse comprometido, en cuyo caso sólo proporciona una falsa sensación de seguridad (lo que es peor que no disponer de cortafuegos).

Cisco y Check Point son dos de las empresas líderes actuales de distribución de cortafuegos. También puede crearse fácilmente un cortafuegos (filtro de paquetes) a partir de una máquina Linux utilizando iptables (software de dominio público, que se suministra habitualmente con Linux).

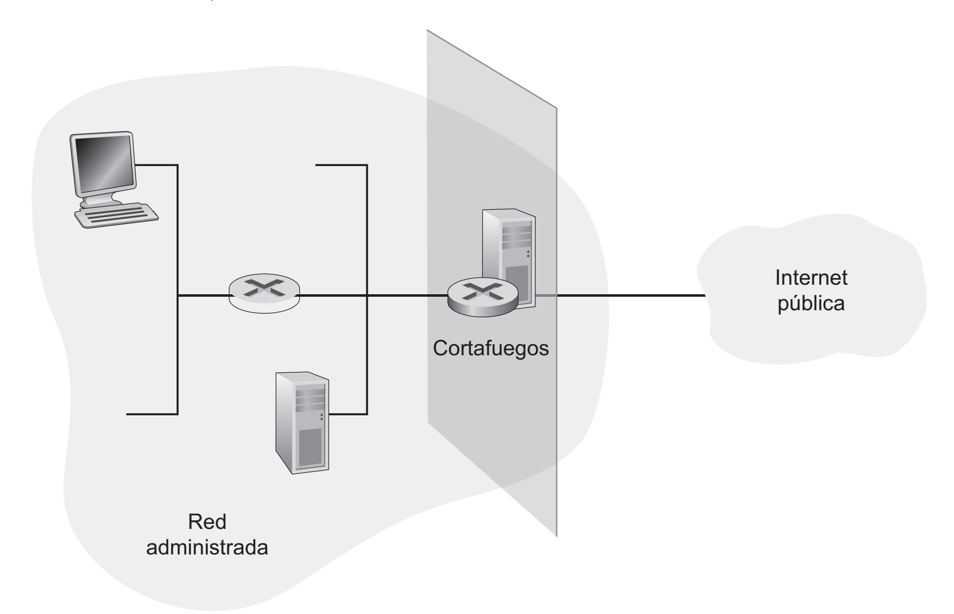


Figura 8.34 • Colocación de un cortafuegos entre la red administrada y el mundo exterior.

Los cortafuegos se pueden clasificar en tres categorías: **filtros de paquetes tradicionales**, **filtros con memoria del estado** y **pasarelas de aplicación**. En las siguientes subsecciones abordaremos cada uno de estos tipos de cortafuegos.

Filtros de paquetes tradicionales

Como se muestra en la Figura 8.34, una organización dispone normalmente de un router de pasarela que conecta su red interna con su ISP (y por tanto con la gran red Internet pública). Todo el tráfico que sale y entra en la red interna pasa a través de este router y es en este router donde tiene lugar el **filtrado de paquetes**. Un filtro de paquetes examina cada datagrama aisladamente, determinando si debe pasar o debe ser eliminado basándose en las reglas especificadas por el administrador. Las decisiones de filtrado normalmente se basan en:

- Las direcciones IP de origen o de destino.
- El tipo de protocolo especificado en el campo de datagrama IP: TCP, UDP, ICMP, OSPF, etc.
- El puerto de destino y de origen TCP o UDP.
- Los bits indicadores TCP: SYN, ACK, etc.
- El tipo de mensaje ICMP.
- Diferentes reglas para los datagramas que salen y entran en la red.
- Diferentes reglas para las distintas interfaces del router.

Un administrador de red configura el cortafuegos basándose en la política de la organización. La política puede tener en cuenta la productividad del usuario y el uso del ancho de banda, así como los problemas de seguridad de una organización. La Tabla 8.5 enumera algunas de las posibles políticas que una organización puede tener y cómo podrían controlarse mediante un sistema de filtrado de paquetes. Por ejemplo, si la organización no desea permitir ninguna conexión TCP entrante excepto aquellas destinadas a su servidor web público, puede bloquear todos los segmentos TCP SYN entrantes, salvo aquellos cuyo puerto de destino sea el puerto 80 y cuya dirección IP de destino sea la correspondientes al servidor web. Si la organización no desea que sus usuarios monopolicen el ancho de banda de acceso con aplicaciones de radio por Internet, puede bloquear todo el tráfico UDP no crítico (ya que la radio por Internet a menudo se transmite sobre UDP). Si la organización no desea que su red interna sea analizada (con Traceroute) por alguien externo, puede bloquear todos los mensajes ICMP TTL caducados que salen de la red de la organización.

Una política de filtrado puede estar basada en una combinación de direcciones y números de puerto. Por ejemplo, un router de filtrado podría reenviar todos los datagramas Telnet (aquellos con un número de puerto igual a 23) excepto aquellos que se dirigen y proceden de una dirección incluida en una lista de direcciones IP especificadas. Esta política permite las conexiones Telnet dirigidas y procedentes de los hosts especificados en la lista de direcciones permitidas. Lamentablemente, basar la política en direcciones externas no proporciona ninguna protección frente a datagramas en los que su dirección de origen haya sido falsificada.

El filtrado también puede basarse en si se ha configurado o no el bit TCP ACK. Este truco resulta bastante útil si una organización desea permitir que sus clientes internos se conecten a servidores externos, pero quiere impedir que clientes externos se conecten a los servidores internos. Recuerde de la Sección 3.5 que el primer segmento de toda conexión TCP tiene puesto a 0 el bit ACK, mientras que los restantes segmentos de la conexión tienen

Política	Configuración del cortafuegos
Sin acceso web externo.	Eliminar todos los paquetes salientes hacia cualquier dirección IP, puerto 80.
Sin conexiones TCP entrantes, excepto las destinadas al servidor web público de la organización.	Eliminar todos los paquetes TCP SYN entrantes hacia cualquier IP excepto 130.207.244.203, puerto 80.
Impedir que las aplicaciones de radio web consuman el ancho de banda disponible.	Eliminar todos los paquetes UDP entrantes, excepto los paquetes DNS.
Impedir que la red sea utilizada para llevar a cabo un ataque DoS.	Eliminar todos los paquetes ping ICMP hacia una dirección de "difusión" (por ejemplo, 130.207.255.255).
Impedir que la red sea examinada con Traceroute	Eliminar todo el tráfico ICMP TTL saliente caducado.

Tabla 8.5 • Políticas y reglas de filtrado correspondientes de la red 130.27/16 de una organización con un servidor web en 130.207.244.203.

puesto a 1 dicho bit. Por tanto, si una organización desea impedir que los clientes externos inicien conexiones con los servidores internos, basta con filtrar todos los segmentos entrantes que tengan el bit ACK puesto a 0. Esta política prohibe todas las conexiones TCP que tienen su origen en el exterior, pero permite las conexiones originadas internamente.

Las reglas de cortafuegos se implementan en los routers mediante listas de control de acceso, teniendo cada interfaz del router su propia lista. En la Tabla 8.6 se muestra un ejemplo de una lista de control de acceso para una organización 222.22/16. Esta lista de control de acceso es para una interfaz que conecta el router con los ISP externos a la organización. Las reglas se aplican a cada datagrama que atraviesa la interfaz de arriba a abajo. Las dos primeras reglas juntas permiten a los usuarios internos navegar por la Web: la primera regla permite salir de la red de la organización a cualquier paquete TCP con el puerto de destino 80; la segunda regla permite entrar en la red de la organización a cualquier paquete TCP que tenga el puerto de origen 80 y el bit ACK activado. Observe que si un origen externo intenta establecer una conexión TCP con un host interno la conexión será bloqueada, incluso aunque el puerto de origen o el de destino sea el puerto 80. Las dos reglas siguientes juntas permiten a los paquetes DNS entrar y salir de la red de la organización. En resumen, esta lista de control de acceso bastante restrictiva bloquea todo el tráfico excepto el tráfico web iniciado dentro de la organización y el tráfico DNS. [CERT Filtering 2009] proporciona una lista de filtros de paquetes basados en puertos/protocolos recomendados para evitar una serie de agujeros de seguridad bien conocidos en las aplicaciones de red existentes.

Filtros de paquetes con memoria del estado

En un filtro de paquetes tradicional, las decisiones de filtrado se toman para cada paquete de forma aislada. Los filtros con memoria del estado realmente controlan las conexiones TCP y utilizan dicha información para tomar las decisiones de filtrado.

Para entender los filtros con memoria del estado, examinemos de nuevo la lista de control de acceso de la Tabla 8.6. Aunque bastante restrictiva, la lista de control de acceso de esta tabla permite que cualquier paquete procedente del exterior con ACK = 1 y puerto de

Acción	Dirección de origen	Dirección de destino	Protocolo	Puerto de origen	Puerto de destino	Bit indicador
Permitir	222.22/16	fuera de 222.22/16	TCP	> 1023	80	cualquiera
Permitir	fuera de 222.22/16	222.22/16	TCP	80	> 1023	ACK
Permitir	222.22/16	fuera de 222.22/16	UDP	> 1023	53	—
Permitir	fuera de 222.22/16	222.22/16	UDP	53	> 1023	—
Denegar	todos	todos	todos	todos	todos	todos

Tabla 8.6 • Lista de control de acceso para una interfaz de router.

origen 80 atraviese el filtro. Tales paquetes podrían ser utilizados por posibles atacantes para intentar hacer fallar a los sistemas internos con paquetes mal formados, llevar a cabo ataques de denegación de servicio o realizar un mapa de la red interna. La solución más sencilla consiste en bloquear también los paquetes TCP ACK, pero este método impediría a los usuarios internos de la organización navegar por la Web.

Los filtros con memoria del estado resuelven este problema almacenando la información de todas las conexiones TCP activas en una tabla de conexiones. Esto es posible porque el cortafuegos puede observar el inicio de una nueva conexión observando un acuerdo en tres fases (SYN, SYNACK y ACK) y puede observar el fin de una conexión cuando ve un paquete FIN para la conexión. El cortafuegos también puede suponer (de forma conservadora) que la conexión ha terminado cuando no ha observado ninguna actividad en la misma durante, por ejemplo, 60 segundos. En la Tabla 8.7 se muestra un ejemplo de una tabla de conexiones para un cortafuegos. Esta tabla de conexiones indica que actualmente hay tres conexiones TCP activas, habiéndose iniciado todas ellas dentro de la organización. Adicionalmente, el filtro con memoria del estado incluye una nueva columna, "Comprobar conexión", en su lista de control de acceso, como se muestra en la Tabla 8.8. Observe que esta tabla es idéntica a la lista de control de acceso de la Tabla 8.6, excepto en que ahora indica que la conexión debería ser comprobada para dos de las reglas.

Veamos algunos ejemplos para examinar cómo trabajan conjuntamente la tabla de conexiones y la lista de control de acceso ampliada. Suponga que un atacante intenta introducir un paquete mal formado en la red de la organización enviando un datagrama con el puerto de origen TCP número 80 y con el bit indicador ACK activado. Suponga además que este paquete tiene el número de puerto de origen 12543 y la dirección IP de origen 150.23.23.155. Cuando este paquete llega al cortafuegos, éste comprueba la lista de control de acceso de la Tabla 8.8, la cual indica que la tabla de conexiones también tiene que ser comprobada antes de permitir la entrada a este paquete en la red de la organización. El cortafuegos comprueba debidamente la tabla de conexiones, ve que ese paquete no es parte de una conexión TCP activa y lo rechaza. Como segundo ejemplo, suponga que un usuario interno desea navegar por un sitio web externo. Puesto que este usuario en primer lugar envía un segmento TCP SYN, la conexión TCP del usuario se registra en la tabla de conexiones. Cuando el servidor web devuelve los paquetes (con el bit ACK necesariamente activado), el cortafuegos comprueba la tabla y ve que la correspondiente conexión está en curso. Por tanto, el cortafuegos deja pasar a estos paquetes, no interviniendo entonces en la actividad de navegación por la Web del usuario interno.

Dirección de origen	Dirección de destino	Puerto de origen	Puerto de destino
222.22.1.7	37.96.87.123	12699	80
222.22.93.2	199.1.205.23	37654	80
222.22.65.143	203.77.240.43	48712	80

Tabla 8.7 • Tabla de conexiones de un filtro con memoria del estado.

Acción	Dirección de origen	Dirección de destino	Protocolo	Puerto de origen	Puerto de destino	Bit indicador	Comprobar conexión
Permitir	222.22/16	fuera de 222.22/16	TCP	> 1023	80	cualquiera	
Permitir	fuera de 222.22/16	222.22/16	TCP	80	> 1023	ACK	X
Permitir	222.22/16	fuera de 222.22/16	UDP	> 1023	53	—	
Permitir	fuera de 222.22/16	222.22/16	UDP	53	> 1023	—	X
Denegar	todos	todos	todos	todos	todos	todos	

Tabla 8.8 • Lista de control de acceso para un filtro con memoria del estado.

Pasarela de aplicación

En los ejemplos anteriores hemos visto que el filtrado en el nivel de paquetes permite a una organización realizar un filtrado basto, basándose en los contenidos de las cabeceras IP y TCP/UDP, incluyendo las direcciones IP, los números de puerto y los bits de reconocimiento (ACK). Pero, ¿qué ocurre si una organización desea proporcionar un servicio Telnet a un conjunto restringido de usuarios internos (en oposición a direcciones IP)? ¿Y qué ocurre si la organización desea que tales usuarios privilegiados se autentiquen a sí mismos antes de permitirles establecer conexiones Telnet con el mundo exterior? Tales tareas quedan fuera de las capacidades de los filtros tradicionales y con memoria del estado. De hecho, la información acerca de la identidad de los usuarios internos está en los datos de la capa de aplicación y no está incluida en las cabeceras IP/TCP/UDP.

Para conseguir seguridad con un nivel de granularidad más fino, los cortafuegos deben combinar los filtros de paquetes con pasarelas de aplicación. Las pasarelas de aplicación miran más allá de la cabeceras IP/TCP/UDP y toman sus decisiones basándose en los datos de aplicación. Una **pasarela de aplicación** es un servidor específico de aplicación a través del cual deben pasar todos los datos de aplicación (entrantes y salientes). Pueden ejecutarse varias aplicaciones de pasarela sobre el mismo host, pero cada pasarela es un servidor separado con sus propios procesos.

Con el fin de proporcionar algunas nociones sobre las pasarelas de aplicación, vamos a diseñar un cortafuegos que permita sólo a un grupo restringido de usuarios internos establecer una conexión Telnet con el exterior y que impida a todos los clientes externos establecer una conexión Telnet con la red interna. Tal política puede conseguirse implementando una combinación de un filtro de paquetes (en un router) y una pasarela de aplicación Telnet, como se muestra en la Figura 8.35. El filtro del router está configurado para bloquear todas las conexiones Telnet excepto aquellas que tienen su origen en la dirección IP de la pasarela de aplicación. Una configuración de filtro como ésta fuerza a todas las conexiones Telnet salientes a pasar a través de la pasarela de aplicación. Considere ahora un usuario interno que quiera conectarse mediante Telnet con el mundo exterior. El usuario tiene que establecer en primer lugar una sesión Telnet con la pasarela de aplicación. Una aplicación que se ejecute en la pasarela y que está a la escucha de sesiones Telnet entrantes pedirá al usuario que introduzca un ID de usuario y una contraseña. Cuando el usuario proporcione esta información, la pasarela de aplicación la comprobará para ver si el usuario tiene permiso para establecer una conexión Telnet con el mundo exterior. Si no es así, la pasarela termina la conexión Telnet que tiene con el usuario interno. Si el usuario tiene permiso, entonces la pasarela (1) pide al usuario el nombre del host externo con el que desea conectarse, (2) establece una sesión Telnet entre la pasarela y el host externo y (3) reenvía al host externo todos los datos que lleguen del usuario, de la misma manera que reenvía el usuario todos los datos que lleguen desde el host externo. De este modo, la pasarela de aplicación Telnet no sólo se encarga de realizar la autorización del usuario, sino que actúa como servidor y cliente Telnet, retransmitiendo la información entre el usuario y el servidor Telnet remoto. Observe que el filtro permitirá que se lleve a cabo el paso 2, dado que es la pasarela quien inicia la conexión Telnet con el mundo exterior.

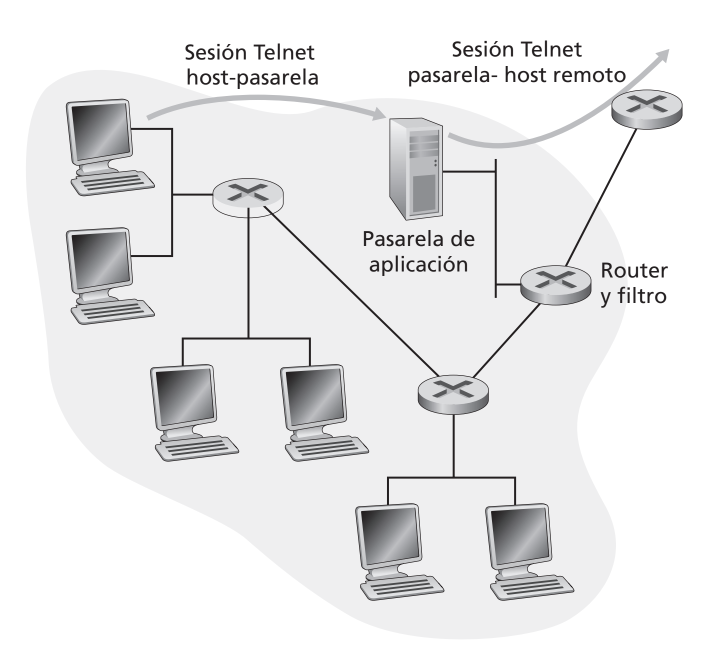
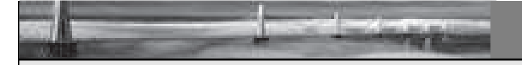


Figura 8.35 • Cortafuegos que consta de una pasarela de aplicación y un filtro.

Las redes internas a menudo disponen de múltiples pasarelas de aplicación, por ejemplo, pasarelas para Telnet, HTTP, FTP y correo electrónico. De hecho, el servidor de correo de una organización (véase la Sección 2.4) y la caché web son pasarelas de aplicación.

Pero las pasarelas de aplicación también tienen sus desventajas. En primer lugar, hace falta una pasarela de aplicación diferente para cada aplicación. En segundo lugar, el rendimiento se verá afectado, dado que todos los datos tendrán que ser reenviados a través de la pasarela. Esto se convierte en un problema particularmente en aquellos casos en los que hay



HISTORIA

ANONIMATO Y PRIVACIDAD

Suponga que deseamos visitar un sitio web controvertido (por ejemplo, un sitio de activismo político) y que (1) no queremos revelar nuestra dirección IP al sitio web, (2) no queremos que el ISP local (que podría ser el ISP de nuestro domicilio o empresa) sepa que estamos visitando dicho sitio y (3) no deseamos que el ISP local vea los datos que estamos intercambiando con el sitio. Si se emplea el método tradicional de conexión directa con el sitio web sin ningún tipo de cifrado no será posible conseguir ninguno de los tres objetivos anteriores. Incluso utilizando SSL los dos primeros objetivos serán inalcanzables: la dirección IP de origen será presentada al sitio web en cada datagrama que envíemos y la dirección de destino de cada paquete que envíemos podrá ser fácilmente husmeada por el ISP local.

Para conseguir la confidencialidad y el anonimato, podemos en su lugar utilizar una combinación de servidor proxy de confianza y SSL, como se muestra en la Figura 8.36. Con esta técnica, primero establecemos una conexión SSL con el proxy de confianza. Después enviamos a través de esta conexión SSL una solicitud HTTP de una página situada en el sitio web deseado. Cuando el proxy reciba la solicitud HTTP cifrada mediante SSL la descifrará y reenviará la solicitud HTTP de texto en claro al sitio web. El sitio web responderá entonces al proxy que a su vez nos reenviará la respuesta a través de SSL. Puesto que el sitio web sólo ve la dirección IP del proxy, y no la dirección de nuestro cliente, estaremos obteniendo un acceso verdaderamente anónimo al sitio web. Y como todo el tráfico entre nosotros y el proxy está cifrado, nuestro ISP local no puede invadir nuestra intimidad almacenando en un registro el nombre del sitio que hemos visitado o los datos que estamos intercambiando. Muchas empresas ofrecen en la actualidad tales servicios proxy (como por ejemplo proxify.com).

Por supuesto, con esta solución, nuestro proxy conoce todos los datos: conoce nuestra dirección IP y la dirección IP del sitio que estamos navegando; y puede ver como texto en claro todo el tráfico intercambiado entre nosotros y el sitio web. Dicha solución, por tanto, sólo será buena si nuestra confianza en el proxy es alta. Un enfoque más robusto adoptado por el servicio de anonimato y privacidad TOR consiste en enrutar el tráfico a través de una serie de servidores proxy no confabulados [TOR 2009]. En particular, TOR permite a los usuarios sugerir nuevos proxies para su lista. Cuando un usuario se conecta a un servidor utilizando TOR, este servicio selecciona aleatoriamente (de entre su lista de proxies) una cadena de tres proxies y enruta todo el tráfico entre el cliente y el servidor a través de esa cadena. De esta forma, suponiendo que los proxies no estén confabulados, nadie sabe que ha tenido lugar una comunicación entre nuestra dirección IP y el sitio web objetivo. Además, aunque la comunicación se realice como texto en claro entre el último proxy y el servidor, el último proxy no sabe qué dirección IP está enviando y recibiendo ese texto en claro.

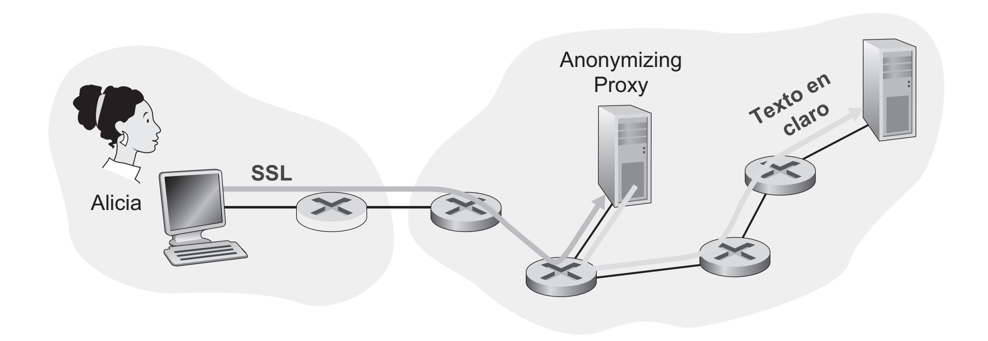


Figura 8.36 • Cómo proporcionar anonimato y privacidad con un proxy.

varios usuarios o aplicaciones utilizando la misma máquina pasarela. Finalmente, el software de cliente debe saber cómo contactar con la pasarela cuando el usuario realiza una solicitud, y debe saber también cómo decir a la pasarela de aplicación con qué servidor externo hay que conectarse.

8.8.2 Sistemas de detección de intrusiones

Ya hemos visto que un filtro de paquetes (tradicional y con memoria del estado) inspecciona los campos de cabecera TCP, UDP e ICMP a la hora de decidir qué paquetes deben pasar a través del cortafuegos. Sin embargo, para detectar muchos tipos de ataques necesitamos llevar a cabo una **inspección profunda de paquetes,** es decir, mirar más allá de los campos de cabecera examinando los propios datos de aplicación transportados por los paquetes. Como hemos visto en la Sección 8.8.1, las pasarelas de aplicación realizan a menudo una inspección profunda de los paquetes. Pero cada pasarela de aplicación sólo lleva a cabo esa tarea para una aplicación específica.

Por ello, existirá un nicho para otro tipo más de dispositivo: un dispositivo que no sólo examine las cabeceras de todos los paquetes que le atraviesen (como los filtros de paquetes), sino que también lleve a cabo una inspección profunda de los paquetes (a diferencia de lo que sucede con los filtros de paquetes). Cuando uno de tales dispositivos detecte un paquete sospechoso o una serie sospechosa de paquetes puede impedir que esos paquetes entren en la red de la organización. O, si la actividad sólo se considera sospechosa, el dispositivo podría dejar pasar los paquetes pero enviando alertas a un administrador de red, que puede de ese modo echar un vistazo más detallado a dicho tráfico y tomar las medidas oportunas. Un dispositivo que genere alertas cuando observe la presencia de tráfico potencialmente malicioso se denomina **Sistema de detección de intrusiones (IDS, *Intrusion Detection System*)**. Un dispositivo que filtre el tráfico sospechoso se denomina **Sistema de prevención de intrusiones (IPS, *Intrusion Prevention System*)**. En esta sección vamos a estudiar ambos tipos de sistemas (IDS e IPS) conjuntamente, dado que el aspecto técnico más interesante de dichos sistemas es cómo detectan el tráfico sospechoso (y no si envían alertas o eliminan los paquetes). Por tanto, vamos a utilizar el término sistemas IDS para referirnos tanto a los sistemas IPS como a los IDS.

Un sistema IDS puede emplearse para detectar una amplia gama de ataques, incluyendo los de mapeado de red (generados, por ejemplo, por nmap), escaneo de puertos, escaneos de

la pila TCP, ataques DoS de inundación del ancho de banda, gusanos y virus, ataques de vulnerabilidades del sistema operativo y ataques de vulnerabilidades de aplicación. (En la Sección 1.6 puede ver una panorámica de los ataques de red.) Actualmente miles de organizaciones utilizan sistemas IDS. Muchos de estos sistemas implementados son propietarios, comercializados por Cisco, Check Point y otros fabricantes de equipos de seguridad. Pero otros muchos de los sistemas IDS implantados son sistemas de dominio público, como el tremendamente popular sistema IDS Snort del que hablaremos en breve.

Un organización puede implantar uno o más sensores IDS en su red. La Figura 8.37 muestra una organización con tres sensores IDS. Cuando se implantan múltiples sensores, normalmente funcionan de manera concertada, enviando información acerca de las actividades de tráfico sospechoso a un procesador IDS central que recopila e integra la información y envía alarmas a los administradores de la red cuando lo considera apropiado. En la Figura 8.37, la organización ha dividido su red en dos regiones: una región de alta seguridad, protegida por un filtro de paquetes y una pasarela de aplicación y monitorizada por sensores IDS; y una región de menor seguridad, denominada **Zona desmilitarizada (DMZ, *Demilitarized Zone*)** que está protegida sólo por el filtro de paquetes, aunque también está monitorizada mediante sensores IDS. Observe que la DMZ incluye los servidores de la organización que necesitan comunicarse con el mundo exterior, como su servidor web público y su servidor DNS autoritativo.

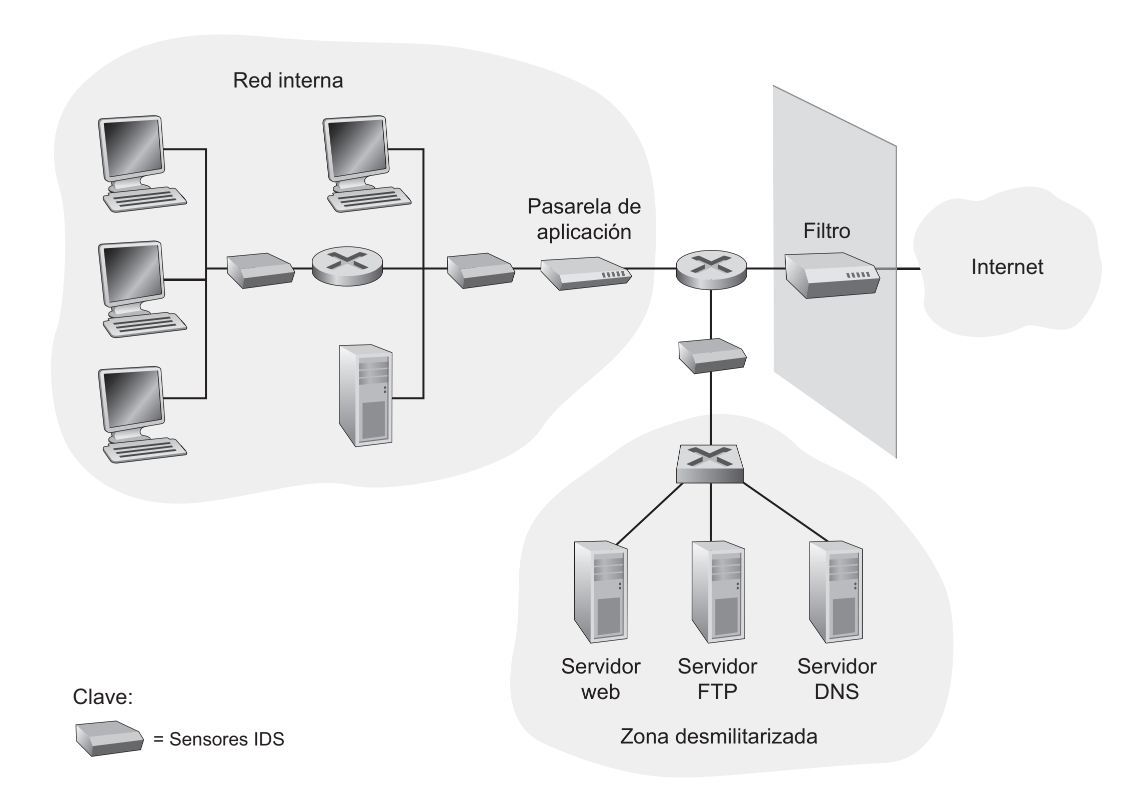


Figura 8.37 • Una organización en la que se ha implantado un filtro, una pasarela de aplicación y sensores IDS.

El lector puede estar preguntándose al llegar a este punto por qué se emplean múltiples sensores IDS. ¿Por qué no colocar simplemente un sensor IDS justo detrás del filtro de paquetes (o incluso integrado con el filtro de paquetes) en la Figura 8.37? Como vamos a ver en breve, un IDS no sólo tiene que llevar a cabo una inspección profunda de los paquetes, sino que también debe comparar cada paquete que pasa con decenas de miles de "firmas"; esto puede requerir una cantidad significativa de procesamiento, en particular si la organización recibe del orden de gigabits/segundo de tráfico procedente de Internet. Colocando los sensores IDS un poco más aguas abajo, cada sensor sólo ve una fracción del tráfico de la organización y puede cumplir más fácilmente con su tarea. De todos modos, hoy día hay disponibles sistemas IDS e IPS de altas prestaciones y muchas organizaciones suelen conformarse con un único sensor localizado cerca de su router de acceso.

Los sistemas IDS se pueden clasificar en términos generales en **sistemas basados en firma** o **sistemas basados en anomalías**. Un IDS basado en firmas mantiene una amplia base de datos de firmas de ataque. Cada firma es un conjunto de reglas concernientes a una actividad de intrusión. Una firma puede ser simplemente una lista de características acerca de un determinado paquete (por ejemplo, números de puerto de origen y de destino, tipo de protocolo y una cadena específica de bits en la carga útil del paquete) o puede estar relacionada con una serie de paquetes. Las firmas normalmente son creadas por ingenieros de seguridad de red experimentados que se dedican a investigar los ataques conocidos. El administrador de red de una organización puede personalizar las firmas o añadir otras de su creación a la base de datos.

Operacionalmente, un sistema IDS basado en firmas analiza cada paquete que pasa a su través, comparando cada paquete husmeado con las firmas de su base de datos. Si un paquete (o una serie de paquetes) concuerda con una firma de la base de datos, el IDS genera una alerta. La alerta podría enviarse al administrador de red en un mensaje de correo electrónico, podría mandarse al sistema de gestión de la red o podría simplemente almacenarse en un registro para su futura inspección.

Los sistemas IDS basados en firma, aunque están ampliamente implantados, presentan una serie de limitaciones. La más importante es que se requiere un conocimiento previo del ataque para generar una firma precisa. En otras palabras, un IDS basado en firmas es completamente inútil frente a nuevos ataques que todavía no hayan sido investigados. Otra desventaja es que incluso si se produce una concordancia con una firma, puede que dicha concordancia no sea el resultado de un ataque, con lo que se generaría una falsa alarma. Finalmente, puesto que es necesario comparar cada paquete con una amplia colección de firmas, el IDS puede verse desbordado por las necesidades de procesamiento y debido a ello fracasar a la hora de detectar muchos paquetes maliciosos.

Un IDS basado en anomalías crea un perfil de tráfico observando el tráfico durante la operación normal. Después busca flujos de paquetes que sean estadísticamente inusuales, como por ejemplo un porcentaje inusual de paquetes ICMP o un crecimiento exponencial súbito en el escaneo de puertos y barridos mediante ping. Lo mejor de los sistemas IDS basados en anomalías es que no dependen del conocimiento previo acerca de los ataques existentes; es decir, pueden detectar potencialmente nuevos ataques no documentados. Por otro lado, el problema de distinguir entre el tráfico normal y el tráfico estadísticamente inusual es extremadamente complejo. A fecha de hoy, la mayoría de los sistemas IDS implantados son basados en signaturas, aunque algunos de ellos incluyen algunas características de los sistemas basados en anomalías.

Snort

Snort es un IDS de dominio público y código abierto, con centenares de miles de implantaciones conocidas [Snort 2007; Koziol 2003]. Puede ejecutarse sobre plataformas Linux, UNIX y Windows. Utiliza la interfaz genérica de análisis libpcap, que también es utilizada por Wireshark y otros muchos husmeadores de paquetes. Puede gestionar fácilmente 100 Mbps de tráfico; para instalaciones con velocidades de tráfico del orden de gibabit/segundo puede ser necesario emplear múltiples sensores Snort.

Para entender un poco cómo funciona Snort, examinemos un ejemplo de signatura utilizada por Snort:

```
alert icmp $EXTERNAL_NET any -> $HOME_NET any
(msg:"ICMP PING NMAP"; dsize: 0; itype: 8;)
```

Esta signatura concordará con cualquier paquete ICMP que entre en la red de la organización (`$HOME_NET`) desde el exterior (`$EXTERNAL_NET`), que sea de tipo 8 (ping ICMP) y que carezca de carga útil (dsize = 0). Puesto que nmap (véase la Sección 1.6) genera paquetes ping con estas características específicas, esta signatura está diseñada para detectar los barridos ping realizados con nmap. Cuando un paquete concuerda con esta signatura, Snort genera una alerta que incluye el mensaje `"ICMP PING NMAP"`.

Lo más impresionante acerca de Snort quizá sea la enorme comunidad de usuarios y expertos de seguridad que mantienen su base de datos de signaturas. Normalmente, al cabo de pocas horas de detectarse un ataque la comunidad Snort escribe y publica una signatura del ataque que después es descargada por los centenares de miles de implantaciones Snort distribuidas por todo el mundo. Además, utilizando la sintaxis de signaturas de Snort, los administradores de red pueden adaptar las signaturas a las necesidades de su propia organización modificando las signaturas existentes o creando otras completamente nuevas.

8.9 Resumen

En este capítulo hemos examinado los diversos mecanismos que nuestros amantes secretos, Benito y Alicia, pueden utilizar para comunicarse de forma segura. Hemos visto que Benito y Alicia están interesados en la confidencialidad (en el sentido de que sólo ellos sean capaces de entender el contenido de los mensajes transmitidos), en la autenticación del punto terminal (con el fin de estar seguros de que están hablando entre ellos) y en la integridad de los mensajes (quieren estar seguros de que sus mensajes no son alterados en el camino). Por supuesto, la necesidad de comunicaciones seguras no está confinada a los amantes secretos. De hecho, hemos visto en las Secciones 8.4 a 8.7 que la seguridad puede utilizarse en varias capas de una arquitectura de red para protegerse frente a los malos que tienen en sus manos un enorme arsenal de posibles ataques.

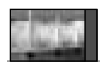
En la primera parte del capítulo hemos presentado varios de los principios que subyacen a las comunicaciones seguras. En la Sección 8.2 hemos cubierto técnicas criptográficas para el cifrado y descifrado de datos, incluyendo la criptografía de clave simétrica y la criptografía de clave pública. DES y RSA se han examinado como casos de estudio específicos de estas dos clases principales de técnicas criptográficas que se emplean en las redes actuales.

En la Sección 8.3 hemos examinado dos métodos que permiten proporcionar mecanismos para asegurar la integridad de los mensajes: los códigos de autenticación de mensajes (MAC, *Message Authentication Code*) y las firmas digitales. Los dos métodos presentan una serie de paralelismos. Ambos utilizan funciones hash criptográficas y ambas técnicas nos permiten verificar el origen del mensaje, así como la integridad del propio mensaje. Una diferencia importante es que los códigos MAC no se basan en el cifrado, mientras que las firmas digitales requieren una infraestructura de clave pública. Ambas técnicas se usan ampliamente en la práctica, como hemos visto en las Secciones 8.4 a 8.7. Además, las firmas digitales se utilizan para crear certificados digitales, los cuales son importantes para verificar la validez de las claves públicas. También hemos investigado la autenticación del punto terminal y hemos visto cómo pueden emplearse los números distintivos para frustrar los ataques por reproducción.

En las Secciones 8.4 a 8.7 hemos examinado varios protocolos de red seguros que disfrutan de un amplio uso en la práctica. Hemos visto que la criptografía de clave simétrica se encuentra en el núcleo de PGP, SSL, IPsec y la seguridad inalámbrica. Hemos visto que la criptografía de clave pública es crucial tanto para PGP como para SSL. También hemos visto que PGP utiliza firmas digitales para proporcionar la integridad de los mensajes, mientras que SSL e IPsec utilizan códigos MAC. Conociendo ahora los principios básicos de la criptografía y habiendo estudiado cómo se utilizan realmente estos principios, el lector está ahora en posición de diseñar sus propios protocolos de red seguros.

Con las técnicas cubiertas en las Secciones 8.2 a 8.7 Benito y Alicia pueden comunicarse de forma segura. (Suponiendo que sean estudiantes de redes que han aprovechado el material de este libro y que de esa forma pueden evitar que Tomás descubra su relación amorosa.) Pero la confidencialidad es sólo una pequeña parte de la panorámica general de la seguridad de las redes. Como hemos señalado en la Sección 8.8, el énfasis en la seguridad de red se ha puesto cada vez más en proteger la infraestructura de la red frente a potenciales ataques realizados por los malos. En la última parte del capítulo hemos abordado los cortafuegos y los sistemas IDS que inspeccionan los paquetes que entran y salen de la red de una organización.

Este capítulo ha cubierto una gran cantidad de temas, centrándose en los aspectos más importantes de la seguridad en las redes modernas. Animamos a los lectores que deseen profundizar a investigar las referencias citadas a lo largo del capítulo. En particular, recomendamos [Skoudis 2006] para los temas relacionados con los ataques a la seguridad operacional, [Kaufman 1995] para el estudio de la criptografía y cómo se aplica a la seguridad de redes, [Rescorla 2001] que proporciona un tratamiento profundo pero asequible sobre SSL y [Edney 2003] que ofrece una exposición sobre la seguridad 802.11, incluyendo una investigación intuitiva sobre WEP y sus desventajas. Los lectores que lo deseen también pueden consultar [Ross 2009] para ver un amplio conjunto de diapositivas PowerPoint (más de 400) sobre la seguridad en las redes, así como varias prácticas de laboratorio basadas en Linux.



Problemas y cuestiones de repaso

Capítulo 8 Problemas de repaso

SECCIÓN 8.1

R1. ¿Cuáles son las diferencias entre la confidencialidad de los mensajes y la integridad de los mismos? ¿Puede existir confidencialidad sin integridad? ¿Puede existir integridad sin confidencialidad? Justifique su respuesta.

R2. Las entidades Internet (routers, dispositivos de conmutación, servidores DNS, servidores web, sistemas terminales de usuario, etc.) a menudo necesitan comunicarse de forma segura. Proporcione tres parejas específicas de ejemplo de entidades de Internet que puedan desear comunicarse de forma segura.

SECCIÓN 8.2

R3. Desde la perspectiva de un servicio, ¿cuál es una diferencia importante entre un sistema de clave simétrica y un sistema de clave pública?

R4. Suponga que un intruso dispone de un mensaje cifrado así como de la versión descifrada de dicho mensaje. ¿Puede el intruso montar un ataque de sólo texto cifrado, un ataque de texto en claro conocido o un ataque de texto claro seleccionado?

R5. Considere un cifrado de 8 bloques. ¿Cuántos posibles bloques de entrada tiene este sistema de cifrado? ¿Cuántas posibles correspondencias existen? Si interpretamos cada correspondencia como una clave, entonces ¿cuántas posibles claves tiene este sistema de cifrado?

R6. Suponga que N personas desean comunicarse con cada una de las otras $N - 1$ personas utilizando un sistema de cifrado de clave simétrica. Todas las comunicaciones entre cualesquiera dos personas, i y j, son visibles para todas las demás personas de ese grupo de N y nadie que no pertenezca a ese grupo debe poder decodificar sus comunicaciones. ¿Cuántas claves son necesarias en el sistema, considerado como un todo? Suponga ahora que se utiliza un sistema de cifrado de clave pública. ¿Cuántas claves se requerirán en este caso?

R7. Suponga que $n = 10.000$, $a = 10.023$ y $b = 10.004$. Utilice una identidad de aritmética modular para calcular mentalmente $(a \bullet b) \bmod n$.

R8. Suponga que desea cifrar el mensaje 10101111 cifrando el número decimal correspondiente al mensaje. ¿Cuál es el número decimal?

SECCIÓN 8.3

R9. ¿De qué forma una función hash proporciona una mejor comprobación de la integridad de los mensajes que una suma de comprobación, tal como la suma de comprobación de Internet?

R10. ¿Se puede “descifrar” un valor hash de un mensaje para obtener el mensaje original? Explique su respuesta.

R11. Considere una variante del algoritmo MAC (Figura 8.9) en la que el emisor envía (m, $H(m) + s$), siendo $H(m) + s$ la concatenación de $H(m)$ y s. ¿Tiene algún defecto esta variante? ¿Por qué?

R12. ¿Qué quiere decir que un documento firmado sea verificable y no falsificable?

R13. ¿En qué sentido es mejor utilizar como firma digital la versión cifrada mediante clave pública del valor hash de un mensaje, en lugar de usar la versión cifrada mediante clave pública del mensaje completo?

R14. Suponga que certificador.com crea un certificado para foo.com. Normalmente, el certificado completo será cifrado con la clave pública de certificador.com. ¿Verdadero o falso?

R15. Suponga que Alicia tiene un mensaje que está dispuesta a enviar a cualquiera que se lo solicite. Miles de personas desean obtener el mensaje de Alicia, pero cada una de ellas desea estar segura de la integridad del mensaje. En este contexto, ¿qué piensa que es más adecuado un esquema de integridad basado en MAC o uno basado en firma digital? ¿Por qué?

R16. ¿Cuál es el propósito de un número distintivo en un protocolo de autenticación de punto terminal?

R17. ¿Qué quiere decir que un número distintivo es un valor que sólo se usa una vez en la vida? ¿En la vida de quién?

R18. ¿Qué es un ataque por interposición? ¿Puede producirse este ataque cuando se utilizan claves simétricas?

SECCIONES 8.4–8.7

R19. Suponga que Benito recibe un mensaje PGP de Alicia. ¿Cómo puede Benito estar seguro de que Alicia ha creado el mensaje (en lugar de, por ejemplo, Tomás)? ¿Utiliza PGP un valor MAC para garantizar la integridad del mensaje?

R20. En el registro SSL existe un campo para los números de secuencia SSL. ¿Verdadero o falso?

R21. ¿Cuál es el propósito de los números distintivos aleatorios en el proceso de acuerdo de SSL?

R22. Suponga que una sesión SSL emplea un cifrado de bloque con CBC. Verdadero o falso: el servidor envía al cliente el vector de inicialización (IV) en claro.

R23. Suponga que Benito inicia una conexión TCP con Tomás, que pretende hacerse pasar por Alicia. Durante la fase de acuerdo, Tomás envía a Benito el certificado de Alicia. ¿En qué paso del algoritmo de acuerdo SSL descubrirá Benito que no está comunicándose con Alicia?

R24. Considere la transmisión de un flujo de paquetes desde el host A al host B utilizando IPsec. Normalmente se establecerá una nueva asociación de seguridad (SA) para cada paquete enviado del flujo. ¿Verdadero o falso?

R25. Suponga que TCP está ejecutándose sobre IPsec entre las sucursales y la oficina principal de la Figura 8.29. Si TCP retransmite el mismo paquete, entonces los dos paque-

tes correspondientes enviados por R1 tendrán el mismo número de secuencia en la cabecera ESP. ¿Verdadero o falso?

R26. ¿Es lo mismo una asociación de seguridad IKE que una asociación de seguridad IPsec? ¿Verdadero o falso?

R27. Considere el protocolo WEP para 802.11. Suponga que los datos son 10101100 y que el flujo de claves es 1111000. ¿Cuál será el texto cifrado resultante?

R28. En WEP, en cada trama se envía un vector de inicialización (IV) en claro. ¿Verdadero o falso?

SECCIÓN 8.8

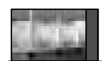
R29. Los filtros de paquetes con memoria del estado mantienen dos estructuras de datos. Nómbrelas y describa brevemente qué hacen.

R30. Considere un filtro de paquetes tradicional (sin conocimiento del estado). Este filtro puede filtrar paquetes basándose en los bits indicadores TCP, así como en otros campos de cabecera. ¿Verdadero o falso?

R31. En un filtro de paquetes tradicional, cada interfaz puede tener su propia lista de control de acceso. ¿Verdadero o falso?

R32. ¿Por qué una pasarela de aplicación debe trabajar conjuntamente con un filtro de router para ser efectiva?

R33. Los sistemas IDS e IPS basados en firma inspeccionan las cargas útiles de los segmentos TCP y UDP. ¿Verdadero o falso?



Problemas

P1. Utilizando el cifrado monoalfabético de la Figura 8.3, codifique el mensaje “Este problema es fácil.” Decodifique el mensaje “rmij’u uamu xyj”.

P2. Demuestre que el ataque de texto en claro conocido de Tomás, en el que Tomás conoce las parejas de traducción (texto cifrado, texto en claro) de doce letras, reduce el número de posibles sustituciones que tienen que comprobarse en el ejemplo de la Sección 8.2.1 en un factor de aproximadamente 10^{15}.

P3. Considere el sistema polialfabético mostrado en la Figura 8.4. Un ataque de texto en claro seleccionado que es capaz de obtener la codificación del texto en claro del mensaje “`Es extraño mojar queso en la cerveza o probar whisky de garrafa`”, ¿será suficiente para decodificar todos los mensajes? ¿Por qué?

P4. Considere el cifrado de bloque de la Figura 8.5. Suponga que cada cifrado de bloque T_i simplemente invierte el orden de los ocho bits de entrada (de manera que, por ejemplo,11110000 se convierte en 00001111). Suponga además que el aleatorizador de 64 bits no modifica ninguno de los bits (de manera que el valor de salida del bit *m-ésimo* es igual al valor de entrada del mismo bit). (a) Con $n = 3$ y siendo la entrada de 64 bits original igual a 10100000 repetido ocho veces, ¿cuál es el valor de salida? (b) Repita el apartado (a) pero ahora cambie el último bit de la entrada de 64 bits original de 0 a

1. (c) Repita los apartados (a) y (b) pero ahora suponga que el aleatorizador de 64 bits invierte el orden de los 64 bits.

P5. Considere el cifrado de bloque de la Figura 8.5. Para una determinada "clave", Alicia y Benito tendrán que mantener ocho tablas, cada una de ellas de 8 bits por 8 bits. Para que Alicia (o Benito) pueda almacenar las ocho tablas, ¿cuántos bits de almacenamiento son necesarios? ¿Cómo se compara este número con el número de bits requeridos para un sistema de cifrado de bloque de 64 bits de tabla completa?

P6. Considere el cifrado de bloque de 3 bits de la Tabla 8.1 y que dispone del texto en claro 100100100. (a) Suponga que inicialmente no se utiliza CBC. ¿Cuál es el texto cifrado resultante? (b) Suponga que Tomás husmea el texto cifrado. Suponiendo que Tomás sabe que se está utilizando un cifrado de bloque de 3 bits sin CBC (aunque no sabe cuál es el cifrado específico) ¿qué cosas podrá deducir Tomás? (c) Ahora suponga que se utiliza CBC con IV = 111. ¿Cuál es el texto cifrado resultante?

P7. (a) Utilizando el algoritmo RSA, elija $p = 3$ y $q = 11$, y codifique la palabra "perro" cifrando cada letra por separado. Aplicando el algoritmo de descifrado a la versión cifrada recupere el mensaje de texto en claro original. (b) Repita el apartado (a) pero cifrando ahora la palabra "perro" como un mensaje *m*.

P8. Considere RSA con $p = 5$ y $q = 11$.

a. Calcule el valor de *n* y *z*.

b. Sea *e* igual a 3. ¿Por qué éste es un valor aceptable para *e*?

c. Determine *d* tal que $de = 1 \pmod z$ y $d < 160$.

d. Cifre el mensaje $m = 8$ utilizando la clave (n, e). Sea *c* el texto cifrado correspondiente. Indique los detalles de todos los cálculos realizados. *Sugerencia:* para simplificar los cálculos, utilice la fórmula siguiente:

$$[(a \bmod n) \bullet (b \bmod n)] \bmod n = (a \bullet b) \bmod n$$

P9. En este problema vamos a explorar el algoritmo de cifrado de clave pública de Diffie-Hellman (DH), que permite a dos entidades acordar una clave compartida. El algoritmo de Diffie-Hellman utiliza un número primo grande *p* y otro número también grande *g* pero menor que *p*. Tanto *p* como *g* se hacen públicos (por lo que cualquier atacante podría conocerlos). Con el algoritmo DH, Alicia y Benito seleccionan de forma independiente sendas claves secretas, S_A y S_B, respectivamente. Alicia calcula entonces su clave pública, T_A, elevando *g* a S_A y reduciendo el resultado mod *p*. Benito calcula de forma similar su propia clave pública, T_B, elevando *g* a S_B y tomando luego mod *p*. Alicia y Benito intercambian entonces sus claves públicas a través de Internet. A continuación Alicia calcula la clave secreta compartida *S* elevando T_B a S_A y luego reduciendo a mod *p*. De forma similar, Benito calcula la clave compartida S' elevando T_A a S_B y tomando después mod *p*.

a. Demuestre que, en general, Alicia y Benito obtienen la misma clave simétrica, es decir, demuestre que $S = S'$.

b. Sean $p = 11$ y $g = 2$ y suponga que Alicia y Benito seleccionan sendas claves privadas $S_A = 5$ y $S_B = 12$, respectivamente. Calcule las claves públicas de Alicia y Benito, T_A y T_B. Indique todos los cálculos realizados.

c. Continuando con el apartado (b), calcule ahora la clave simétrica compartida *S*. Indique todos los cálculos realizados.

d. Dibuje un diagrama de temporización que muestre cómo puede atacarse el algoritmo de Diffie-Hellman utilizando un ataque por interposición. El diagrama de temporización debe tener tres líneas verticales: una para Alicia, otra para Benito y una tercera para el atacante Tomás.

P10. Suponga que Alicia quiere comunicarse con Benito utilizando criptografía de clave simétrica con una clave de sesión K_S. En la Sección 8.2 hemos visto cómo puede utilizarse la criptografía de clave pública para distribuir la clave de sesión de Alicia a Benito. En este problema vamos a analizar cómo puede distribuirse la clave de sesión (sin criptografía de clave pública) utilizando un centro de distribución de claves (KDC, *Key Distribution Center*). El KDC es un servidor que comparte una clave simétrica secreta única con cada usuario registrado. Para Alicia y Benito vamos a designar estas claves mediante $K_{A\text{-}KDC}$ y $K_{B\text{-}KDC}$. Diseñe un esquema que utilice el KDC para distribuir K_S a Alicia y a Benito. El esquema debe utilizar tres mensajes para distribuir la clave de sesión: un mensaje de Alicia hacia el KDC, un mensaje del KDC a Alicia y, finalmente, un mensaje de Alicia a Benito. El primer mensajes será $K_{A\text{-}KDC}$ (A, B). Utilizando la notación $K_{A\text{-}KDC}$, $K_{B\text{-}KDC}$, *S, A y B* responda a las siguientes cuestiones:

a. ¿Cuál es el segundo mensaje?

b. ¿Cúal es el tercer mensaje?

P11. Calcule un tercer mensaje diferente de los dos mensajes de la Figura 8.8, que tenga la misma suma de comprobación que los mensajes de la Figura 8.8.

P12. Suponga que Alicia y Benito comparten dos claves secretas: una clave de autenticación S_1 y una clave simétrica de cifrado S_2. Amplíe la Figura 8.9 de forma que se proporcionen tanto integridad como confidencialidad.

P13. En el protocolo de distribución de archivos P2P BitTorrent (véase el Capítulo 2) la semilla descompone el archivo en bloques y los pares se redistribuyen los bloques unos a otros. Si no se utilizara ninguna protección, un atacante podría fácilmente hacer que un torrente dejara de funcionar haciéndose pasar por un par benevolente y enviando bloques falsos a un pequeño subconjunto de pares del torrente. Estos pares no sospechosos redistribuirían entonces los bloques falsos a otros pares, que a su vez los distribuirían a otros. Por tanto, es crítico para el funcionamiento de BitTorrent que el sistema tenga un mecanismo que permita a un par verificar la integridad de un bloque, de modo que no se redistribuyan bloques falsos. Suponga que cuando un par se une a un torrente obtiene inicialmente un archivo `.torrent` de un origen *completamente* de confianza. Describa un esquema simple que permita a los pares verificar la integridad de los bloques.

P14. El protocolo de enrutamiento OSPF utiliza un valor MAC en lugar de firmas digitales para garantizar la integridad de los mensajes. ¿Por qué cree que se eligió un valor MAC en lugar de firmas digitales?

P15. Considere nuestro protocolo de autenticación de la Figura 8.16, en el que Alicia se autentica ante Benito y que hemos visto que funciona bien (es decir, no hemos encontrado en él ningún fallo). Suponga ahora que mientras que Alicia se está autenticando ante Benito, éste debe autenticarse ante Alicia. Indique el escenario mediante el cual

Tomás, haciéndose pasar por Alicia, podría ahora autenticarse ante Benito como si fuera Alicia. (*Sugerencia*: considere que la secuencia de operaciones del protocolo, una en la que Tomás es el iniciador y otra en la que el iniciador es Benito, puede entremezclarse arbitrariamente. Preste atención especial al hecho de que tanto Benito como Alicia utilizan un número distintivo y que, si no se tiene cuidado, alguien podría utilizar maliciosamente el mismo número distintivo.)

P16. En el ataque por interposición (*man-in-the-middle*) de la Figura 8.19 Alicia no ha autenticado a Benito. Si Alicia requiriera a Benito que se autenticara utilizando el protocolo de autenticación de clave pública, ¿podría evitarse el ataque por interposición? Explique su razonamiento.

P17. La Figura 8.20 muestra las operaciones que Alicia debe realizar con PGP para proporcionar confidencialidad, autenticación e integridad de los mensajes. Haga un diagrama de las correspondientes operaciones que Benito debe realizar con el paquete recibido de Alicia.

P18. Suponga que Alicia quiere enviar un correo electrónico a Benito. Benito tiene una pareja de claves pública-privada (K_B^+,K_B^-), y Alicia dispone del certificado de Benito. Pero Alicia no tiene una pareja de claves pública y privada. Alicia y Benito (y todo el mundo) comparten la misma función hash $H(\cdot)$.

a. En esta situación, ¿es posible diseñar un esquema mediante el que Benito pueda verificar que es Alicia quien ha creado el mensaje? En caso afirmativo, indique cómo mediante un diagrama de bloques para Alicia y Benito.

b. ¿Es posible diseñar un esquema que proporcione confidencialidad para enviar el mensaje de Alicia a Benito? En caso afirmativo, indique cómo mediante un diagrama de bloques para Alicia y Benito.

P19. Considere la salida de Wireshark mostrada en la página siguiente para una parte de una sesión SSL.

a. ¿El paquete Wireshark 112 ha sido enviado por el cliente o por el servidor?

b. ¿Cuáles son el número de puerto y la dirección IP del servidor?

c. Suponiendo que no hay pérdidas ni retransmisiones, ¿cuál será el número de secuencia del siguiente segmento TCP enviado por el cliente?

d. ¿Cuántos registros SSL contiene el paquete Wireshark 112?

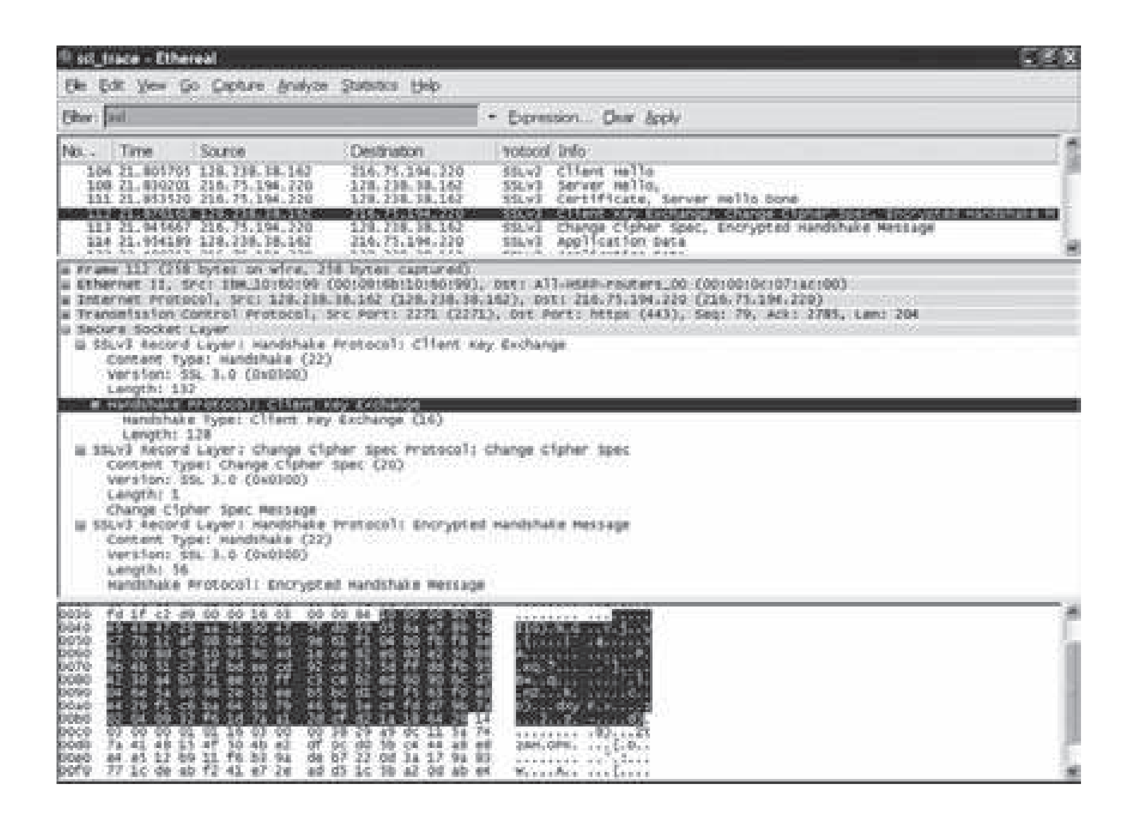
e. El paquete 112 ¿contiene una clave maestra (MS) o una clave maestra cifrada (EMS) o ninguna de las dos?

f. Suponiendo que el campo de tipo de acuerdo es de 1 byte y que cada campo de longitud es de 3 bytes, ¿cuáles son los valores del primer y último byte de la clave maestra (o de la clave maestra cifrada)?

g. ¿Cuántos registros SSL tiene en cuenta el mensaje cifrado de acuerdo del cliente?

h. ¿Cuántos registros SSL tiene en cuenta el mensaje cifrado de acuerdo del servidor?

P20. En la Sección 8.5.1 se muestra que, sin números de secuencia, Tomás (*man-in-the middle*) puede causar una catástrofe en una sesión SSL intercambiando segmentos TCP. ¿Podría Tomás hacer algo similar borrando un segmento TCP? ¿Qué necesitaría hacer para tener éxito con dicho ataque de borrado? ¿Qué efecto tendría?



P21. Suponga que Alicia y Benito se están comunicando mediante una sesión SSL. Suponga que un atacante, que no dispone de ninguna de las claves compartidas, inserta un segmento TCP falso en un flujo de paquetes con la suma de comprobación TCP y los números de secuencia correctos (y con direcciones IP y números de puerto también correctos). ¿Aceptará el SSL en el lado receptor el paquete falso y pasará la correspondiente carga útil a la aplicación receptora? ¿Por qué?

P22. Responda a las siguientes preguntas de tipo verdadero/falso que hacen referencia a la Figura 8.29.

a. Cuando un host de 172.16.1/24 envía un datagrama a un servidor Amazon.com, el router R1 cifrará el datagrama utilizando IPsec.

b. Cuando un host de 172.16.1/24 envía un datagrama a un host de 172.16.2/24, el router R1 cambiará la dirección de origen y de destino del datagrama IP.

c. Suponga que un host de 172.16.1/24 inicia una conexión TCP con un servidor web situado en 172.16.2/24. Como parte de esta conexión, todos los datagramas enviados por R1 tendrán el número de protocolo 50 en el campo de cabecera IPv4 situado más a la izquierda.

d. Considere el envío de un segmento TCP desde un host situado en 172.16.1/24 a un host que se encuentra en 172.16.2/24. Suponga que el reconocimiento de este segmento se pierde, de modo que TCP reenvía el segmento. Puesto que IPsec utiliza números de secuencia, R1 no reenviará el segmento TCP.

P23. Considere el ejemplo de la Figura 8.29. Suponga que Tomás está realizando un ataque por interposición (*man-in-the-middle*) y puede insertar datagramas en el flujo de datagramas que va de R1 a R2. Como parte de un ataque por reproducción, Tomás envía una copia duplicada de uno de los datagramas enviados de R1 a R2. ¿Descifrará R2 el datagrama duplicado y lo reenviará hacia la red de la sucursal? En caso negativo, describa en detalle cómo detecta R2 el datagrama duplicado.

P24. Considere el siguiente protocolo pseudo-WEP. La clave es de 4 bits y el vector IV tiene 2 bits. El vector IV se añade al final de la clave al generar el flujo de claves. Suponga que la clave secreta compartida es 1010. Los flujos de claves para las cuatro entradas posibles son los siguientes:

101000: 0010101101010101001011010100100 . . .

101001: 1010011011001010110100100101101 . . .

101010: 0001101000111100010100101001111 . . .

101011: 1111101010000000101010100010111 . . .

Suponga que todos los mensajes tienen una longitud de 8 bits. Suponga que el ICV (comprobación de integridad) tiene una longitud de 4 bits y que se calcula combinando mediante XOR los primeros 4 bits de datos con los últimos 4 bits de datos. Suponga que el paquete pseudo-WEP consta de tres campos: el primero es el campo IV, luego el campo de mensaje y el último es el campo ICV, estando algunos de estos campos cifrados.

a. Deseamos enviar el mensaje m = 10100000 utilizando el vector IV = 11 y WEP. ¿Cuáles serán los valores de los tres campos WEP?

b. Demuestre que cuando el receptor descifra el paquete WEP, recupera el mensaje y el ICV.

c. Suponga que Tomás intercepta un paquete WEP (no necesariamente con el vector IV = 11) y quiere modificarlo antes de reenviarlo hacia el receptor. Suponga que Tomás invierte el primer bit de ICV. Suponiendo que Tomás no conoce los flujos de claves para ninguno de los vectores IV, ¿qué otro bit o qué otros bits tiene que invertir también Tomás para que el paquete recibido pase la comprobación de ICV?

d. Justifique su respuesta modificando los bits del paquete WEP del apartado (a), descifrando el paquete resultante y verificando que el paquete pasa la comprobación de integridad.

P25. Proporcione una tabla de filtrado y una tabla de conexión para un cortafuegos con memoria del estado que sea lo más restrictivo posible, pero que lleve a cabo lo siguiente:

a. Permitir a todos los usuarios internos establecer sesiones Telnet con hosts externos.

b. Permitir a los usuarios externos navegar por el sitio web de la empresa en la dirección 222.22.0.12.

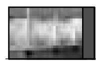
c. En caso contrario, bloquear todos los restantes tipos de tráfico entrante y saliente.

La red interna es 222.22/16. En su solución, suponga que la tabla de conexiones almacena actualmente tres conexiones en caché, todas desde el interior hacia el exterior.

En su solución tendrá que inventar los números de puerto y las direcciones IP apropiados.

P26. Suponga que Alicia desea visitar el sitio web activistas.com utilizando un servicio de tipo TOR. Este servicio utiliza dos servidores proxy no confabulados Proxy1 y Proxy2. Alicia obtiene en primer lugar los certificados (cada uno de ellos contiene una clave pública) para Proxy1 y Proxy2 de algún servidor central. Designaremos mediante $K_1^+(\ )$, $K_2^+(\ )$, $K_1^-(\ )$ y $K_2^-(\ )$ las operaciones de cifrado/descifrado con las claves pública y privada RSA.

a. Utilizando un diagrama de temporización, proporcione un protocolo (lo más simple posible) que permita a Alicia establecer una clave de sesión compartida S_1 con Proxy1. Designe mediante $S_1(m)$ el cifrado/descifrado de los datos m con la clave compartida S_1.
b. Utilizando un diagrama de temporización, proporcione un protocolo (lo más simple posible) que permita a Alicia establecer una clave de sesión compartida S_2 con Proxy2 *sin revelar su dirección IP a Proxy2*.
c. Suponga ahora que las claves compartidas S_1 y S_2 ya están establecidas. Utilizando un diagrama de temporización proporcione un protocolo (lo más simple posible y que *no utilice criptografía de clave pública*) que permita a Alicia solicitar una página html de activistas.com *sin revelar su dirección IP a Proxy2* y *sin revelar a Proxy1 qué sitio está visitando.* El diagrama debe terminar con la llegada de la solicitud HTTP al sitio activistas.com.



Preguntas para la discusión

D1. Suponga que un intruso puede insertar y eliminar mensajes DNS en un servidor DNS. Describa tres escenarios que muestren los problemas que dicho intruso podría causar.

D2. ¿Qué es Kerberos? ¿Cómo funciona? ¿Cómo se relaciona con el Problema P10 de este capítulo?

D3. Si IPsec proporciona seguridad en la capa de red, ¿por qué siguen haciendo falta mecanismos de seguridad en las capas situadas por encima de IP?

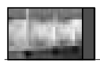
D4. Investigue unos cuantos datos acerca de las botnets. ¿Qué protocolos y sistemas utilizan los atacantes para controlar y actualizar hoy en día las botnets?



Práctica de laboratorio con Wireshark

En esta práctica de laboratorio (disponible en el sitio web del libro) vamos a investigar el protocolo Capa de sockets seguros (SSL, *Secure Sockets Layer*). Recuerde de la Sección 8.5 que SSL se utiliza para dotar de seguridad a una conexión TCP y que se usa ampliamente en la práctica para proporcionar seguridad a las transacciones por Internet. En esta práctica vamos a centrarnos en los registros SSL enviados a través de la conexión TCP. Trataremos de perfilar y clasificar cada uno de los registros, con el objetivo de entender el por qué y el

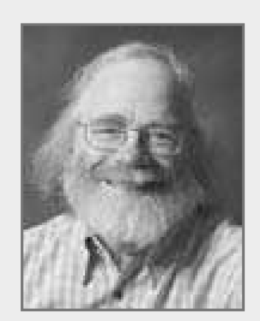
cómo de cada uno de ellos. Investigaremos los diversos tipos de registros SSL, así como los campos de los mensajes SSL. Lo haremos analizando una traza de los registros SSL intercambiados entre nuestro host y un servidor de comercio electrónico.



Práctica de laboratorio con IPsec

En esta práctica de laboratorio (disponible en el sitio web del libro), vamos a explorar cómo crear asociaciones de seguridad IPsec entre máquinas linux. Puede realizar la primera parte de la práctica con dos máquinas linux normales, cada una de las cuales deberá disponer de un adaptador Ethernet. Pero para la segunda parte de la práctica necesitará cuatro máquinas linux, teniendo dos de ellas dos adaptadores Ethernet. En la segunda mitad de la práctica de laboratorio tendrá que crear asociaciones de seguridad IPsec utilizando el protocolo ESP en modo túnel. Tendrá que hacer esto definiendo primero manualmente las asociaciones de seguridad y luego haciendo que IKE las cree.

Steven M. Bellovin



Steven M. Bellovin se unió al claustro de profesores de la Universidad de Columbia después de estar muchos años en el Laboratorio de investigación de servicios de red de AT&T Labs Research en Florham Park, New Jersey, Estados Unidos. Su interés principal se centra en las redes, en la seguridad y en por qué ambos conceptos son incompatibles. En 1995 le concedieron el galardón Usenix Lifetime Achievement Award por su trabajo en al creación de Usenet, la primera red de intercambio de grupos de noticias que permitió enlazar varias computadoras y que los usuarios compartieran información y participaran en discusiones. Steve también es miembro electo de la National Academy of Engineering. Se graduó en la Universidad de Columbia y es doctor por la Universidad de Carolina del Norte en Chapel Hill.

¿Qué le condujo a especializarse en el área de la seguridad de redes?

Puede que le parezca extraño, pero la respuesta es muy simple: era divertido. Mi formación era en programación y administración de sistemas, lo que conduce de una forma bastante natural al campo de la seguridad. Y siempre me han interesado las comunicaciones, desde los tiempos en que realizaba trabajos de programación sobre sistemas en tiempo compartido cuando estudiaba en la universidad.

Mi trabajo en el campo de la seguridad sigue estando motivado por dos cosas: el deseo de continuar haciendo que las computadoras sean útiles, lo que significa que su función no pueda verse corrompida por ningún atacante y el deseo de proteger la privacidad.

¿Cuál era su visión de Usenet mientras la desarrollaba? ¿Y ahora?

Originalmente la veíamos como una forma de hablar acerca de la informática y de la programación de computadoras con otras personas dispersas por todo el país, con el añadido de grupos de usuarios locales para cuestiones administrativas, anuncios de compra-venta, etc. De hecho, mi predicción original es que se iban a generar uno o dos mensajes por día procedentes de entre 50 y 100 sitios como máximo. Pero el crecimiento real experimentado por esa red fue en temas relacionados con la sociedad, incluyendo (pero sin limitarse a ello) la interacción de los seres humanos con las computadoras. Mis grupos de noticias favoritos a lo largo de los años han sido cosas como rec.woodworking (dedicado a la carpintería), así como sci.crypt (dedicado a la criptografía).

Hasta cierto punto, netnews se ha visto desplazado por la Web. Si tuviera que comenzar a rediseñarla hoy en día tendría un aspecto muy distinto. Pero continúa siendo una herramienta excelente para apelar a una audiencia muy amplia interesada en un determinado tema, sin tener que depender de ningún sitio web concreto.

¿Hay alguien que le haya inspirado profesionalmente? ¿En qué forma?

El profesor Fred Brooks (el fundador y director original del departamento de Ciencias de la Computación de la Universidad de Carolina del Norte en Chapel Hill, que era también el jefe del equipo que desarrolló el IBM S/360 y el OS/360, y autor de *The Mythical Man-Month*) tuvo una enorme

influencia en mi carrera profesional. Por encima de todo, me enseñó a mirar las cosas con una perspectiva amplia y a entender los compromisos necesarios: cómo examinar los problemas en el contexto del mundo real (y cuánto más lioso era el mundo real de lo que a un teórico le gustaría) y cómo equilibrar una serie de intereses en conflicto a la hora de diseñar una solución. La mayor parte del trabajo con computadoras es de ingeniería: el arte de llegar a los compromisos adecuados con el fin de satisfacer muchos objetivos contradictorios.

¿Cuál es su visión sobre el futuro de las redes y la seguridad?

Hasta ahora, buena parte de la seguridad de la que disfrutamos procede del aislamiento. Por ejemplo, un cortafuegos funciona cortando el acceso a ciertas máquinas y servicios. Pero nos encontramos en una era de conectividad creciente, con lo que cada vez es más complicado poder aislar las cosas. Y lo que es aún peor es que nuestros sistemas de producción requieren cada vez más elementos separados interconectados mediante redes. Dotar de seguridad a todo esto es uno de nuestros mayores desafíos.

¿Cuáles cree que han sido los mayores avances en el campo de la seguridad? ¿Cuánto más allá debemos ir todavía?

Al menos científicamente sabemos cómo hacer criptografía, lo que ha sido de una gran ayuda. Pero la mayor parte de los problemas de seguridad se deben a los errores en el código y ése es otro problema bastante más complicado. De hecho, es el problema más antiguo aún no resuelto en las ciencias de la computación y, en mi opinión, continuará siendo así. El desafío estriba en concebir cómo dotar de seguridad a los sistemas cuando no tenemos otro remedio que construirlos a partir de elementos inseguros. Hoy día ya podemos hacer eso para conseguir fiabilidad en presencia de fallos hardware; ¿podríamos hacer lo mismo para la seguridad?

¿Qué consejo le daría a los estudiantes relativo a la seguridad de Internet y de las redes en general?

El aprender los mecanismos es la parte más fácil. Lo que es más duro es aprender a pensar "en forma paranoica". Es preciso recordar que las distribuciones de probabilidad no son aplicables en este campo: los atacantes pueden encontrar condiciones improbables y de hecho lo harán. Y los detalles importan; de hecho importan muchísimo.

CAPÍTULO 9

Gestión de redes

Habiendo completado los ocho primeros capítulos de este texto, ya somos perfectamente conscientes de que una red está compuesta por *muchos* elementos complejos de hardware y software interactuando entre sí, desde los enlaces, dispositivos de conmutación, routers, hosts y otros dispositivos que constituyen los componentes físicos de la red, hasta los muchos protocolos (tanto en hardware como en software) que controlan y coordinan estos dispositivos. Cuando una organización interconecta centenares o miles de tales componentes para formar una red, no resulta sorprendente que algunos componentes funcionen en ocasiones de manera inadecuada, que se produzcan fallos de configuración de elementos de la red, que los recursos de la red estén sobrexplotados o que los componentes de la red simplemente se rompan (por ejemplo, alguien puede cortar un cable o derramar una lata de refresco sobre un router). El administrador de la red, cuyo trabajo consiste en mantener la red en funcionamiento, tiene que ser capaz de responder a esos sucesos (o, todavía mejor, de evitarlos). Habiendo potencialmente miles de componentes distribuidos por una extensa área, el administrador de red, que trabaja desde un Centro de operaciones de red (NOC, *Network Operations Center*) necesita obviamente una serie de herramientas que le ayuden a monitorizar, gestionar y controlar la red. En este capítulo vamos a examinar la arquitectura, los protocolos y la base de información usados por los administradores de red para llevar a cabo esta tarea.

9.1 ¿Qué es la gestión de redes?

Antes de profundizar en el tema de la propia gestión de redes, vamos a considerar en primer lugar unos cuantos escenarios ilustrativos del “mundo real”, que no tienen que ver con las redes, pero en los que es necesario que un administrador monitorice, gestione y controle un sistema complejo con muchos componentes interactuando entre sí. Las plantas de genera-

ción de energía eléctrica disponen de una sala de control en la que una serie de diales, medidores e indicadores luminososos monitorizan el estado (temperatura, presión, flujo) de diversas válvulas, conductos y recipientes remotos, así como de otros componentes de la planta. Estos dispositivos permiten al operador monitorizar la multitud de componentes que forman la planta y pueden alertarle (con la famosa luz roja parpadeante de advertencia) cuando esté a punto de surgir un problema. El operador de la planta llevará a cabo una serie de acciones para controlar a estos componentes. De forma similar, la cabina de una aeronave está llena de instrumentos que permiten a un piloto monitorizar y controlar los múltiples componentes que forman un avión. En estos dos ejemplos, el "administrador" *monitoriza* una serie de dispositivos remotos y *analiza* sus datos con el fin de cerciorarse de que esos dispositivos estén operativos y de que su operación se mantenga dentro de una serie de límites prescritos (por ejemplo, asegurándose de que el núcleo de una planta de energía nuclear no se va a fundir de manera inminente, o de que el avión no está a punto de quedarse sin combustible). Asimismo, *controla reactivamente* el sistema haciendo ajustes en respuesta a los cambios que el sistema o su entorno sufren y también *administra proactivamente* el sistema (por ejemplo, detectando tendencias o comportamientos anómalos, lo que permite tomar una serie de medidas antes de que surjan problemas serios). En un sentido similar, el administrador de una red monitorizará, gestionará y controlará activamente el sistema que esté a su cargo.

En los primeros días de las redes, cuando las redes de computadoras eran creaciones producto de la investigación en lugar de una infraestructura crítica utilizada por centenares de millones de personas cada día, el propio concepto de "gestión de red" no existía. Si uno se encontraba con un problema de red, podía ejecutar unos cuantos comandos `ping` para localizar el origen del problema y luego modificar las configuraciones del sistema, reiniciar el hardware o el software o llamar a un colega remoto para que llevara a cabo esas tareas (en [RFC 789] hay disponible una explicación muy amena acerca de la primera caída a gran escala de ARPAnet el 27 de octubre de 1980, mucho antes de que hubiera disponibles herramientas de gestión de red y así como de los esfuerzos que se hicieron para recuperarse de la caída y entender sus motivos). A medida que la red Internet pública y las intranets privadas han ido creciendo, convirtiéndose de pequeñas redes en una gran infraestructura global, ha ido creciendo también la importancia y la necesidad de gestionar el enorme número de componentes hardware y software que forman estas redes.

Como motivación para nuestro estudio de la gestión de redes, vamos a comenzar con un ejemplo simple. En la Figura 9.1 se ilustra una pequeña red compuesta por tres routers y una serie de hosts y servidores. Incluso en una red tan simple como ésta existen muchos escenarios en los que un administrador de red podría aprovechar enormemente la disponibilidad de herramientas de gestión de red:

- *Detección de fallos de una tarjeta de interfaz en un host o un router.* Con las herramientas de gestión de red apropiadas, una entidad de red (por ejemplo, el router A) podría informar al administrador de red que una de sus interfaces ha fallado. (¡Esto es obviamente preferible a recibir en el NOC una llamada de un usuario airado diciendo que la conexión de red no funciona!) Un administrador de red que se dedique activamente a monitorizar y analizar el tráfico de red puede ser capaz de impresionar *realmente* al usuario potencialmente airado, detectando problemas en la interfaz antes de que se produzcan y sustituyendo la tarjeta de interfaz sin esperar a que falle. Esto puede hacerse, por ejemplo, si el administrador de red observa un incremento en los errores de suma de comprobación de las tramas enviadas por esa interfaz que está a punto de morir.

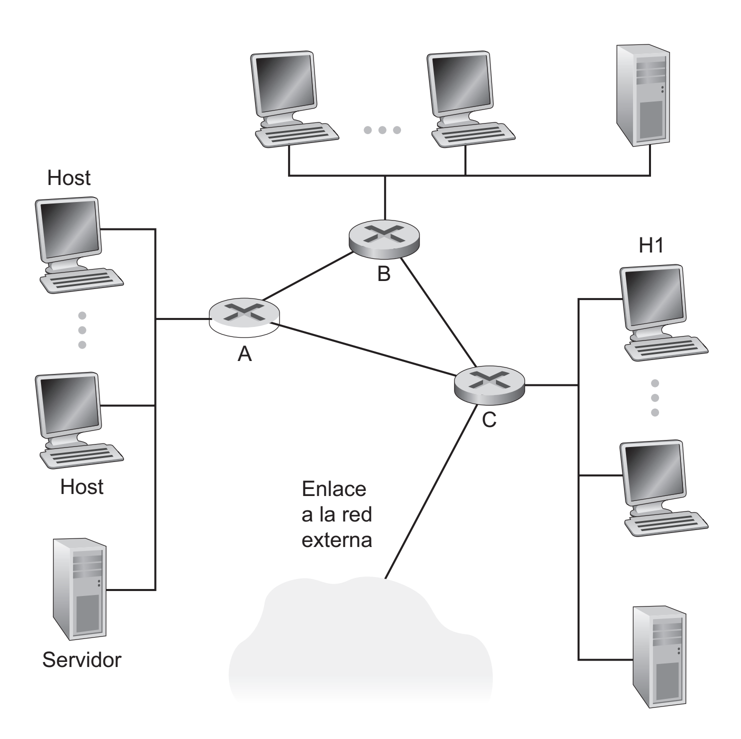


Figura 9.1 • Un escenario simple que ilustra los usos de la gestión de red.

- *Monitorización de los hosts.* El administrador de red puede realizar comprobaciones periódicas para ver si todos los hosts de la red están encendidos y operativos. De nuevo, el administrador de la red puede ser capaz de impresionar a los usuarios respondiendo proactivamente a un problema (la caída de un host) antes de que el propio usuario informe del mismo.
- *Monitorización del tráfico como ayuda a la implantación de recursos.* Un administrador de red puede monitorizar los patrones de tráfico entre cada origen y cada destino y observar, por ejemplo, que si intercambian los servidores entre segmentos de una red LAN la cantidad de tráfico que atraviesa múltiples redes LAN podría reducirse significativamente. Imagine lo feliz que se sentiría todo el mundo cuando se consiguiera así un mejor rendimiento de la red sin necesidad de invertir en nuevos equipos. De forma similar, al monitorizar la utilización de los enlaces el administrador de la red puede determinar que un segmento LAN o que el enlace externo con el resto de mundo está sobrecargado y que sería necesario, en consecuencia, proporcionar un enlace de mayor ancho de banda (por supuesto, con un mayor coste). El administrador de la red también podría querer recibir notificaciones automáticamente cuando los niveles de congestión en un enlace excedan en un valor umbral dado, con el fin de poder instalar un enlace de mayor ancho de banda antes de que la congestión sea demasiado grave.
- *Detección de cambios rápidos en las tablas de enrutamiento.* Las alteraciones rápidas de ruta (cambios frecuentes en las tablas de enrutamiento) pueden indicar inestabilidades en el enrutamiento o la existencia de un router mal configurado. Ciertamente, el administra-

dor de red que haya configurado inadecuadamente un router preferirá descubrir el error por sí mismo antes de que la red se caiga.

- *Monitorización de los SLA.* Los **Acuerdos de nivel del servicio (SLA, *Service Level Agreements*)** son contratos que definen métricas específicas de rendimiento y niveles aceptables de rendimiento del proveedor de la red con respecto a estas métricas [Huston 1999a]. Verizon y Sprint son sólo dos de los muchos proveedores de red que proporcionan acuerdos SLA garantizados a sus clientes [Verizon 2009; Sprint 2009]. Estos SLA incluyen la disponibilidad del servicio (interrupciones), la latencia, la tasa de transferencia y los requisitos de notificación de las interrupciones. Evidentemente, si los criterios de rendimiento han de formar parte de un acuerdo de servicio entre un proveedor de red y sus usuarios, entonces tendrá una gran importancia para el administrador de red ser capaz de medir y gestionar las prestaciones.
- *Detección de intrusiones.* Un administrador de red puede desear que se le notifique que ha llegado un cierto tráfico de red procedente de un origen sospechoso (por ejemplo, de un cierto host o un cierto número de puerto) o con destino a él. De forma similar, el administrador de la red también podría querer detectar (y en muchos casos filtrar) la existencia de ciertos tipos de tráfico (por ejemplo, paquetes con enrutamiento de origen o un gran número de paquetes SYN dirigidos a un cierto host) que se sabe que son característicos de los tipos de ataques que hemos considerado en el Capítulo 8.

La Organización Internacional de Estandarización (ISO) ha creado un modelo de gestión de red que resulta útil para situar los escenarios de ejemplo anteriores dentro de un marco de trabajo más estructurado. Se definen cinco áreas de gestión de la red:

- *Gestión de rendimiento.* El objetivo de la gestión del rendimiento es cuantificar, medir, registrar, analizar y controlar el rendimiento (por ejemplo, la utilización y la tasa de transferencia) de distintos componentes de la red. Entre estos componentes se incluyen dispositivos individuales (como por ejemplo, enlaces, routers y hosts), así como abstracciones terminal a terminal, tales como una ruta a través de la red. Como veremos enseguida, hay ciertos protocolos estándar, como el Protocolo simple de gestión de red (SNMP, *Simple Network Management Protocol*) [RFC 3410], que desempeñan un papel fundamental en la gestión del rendimiento de Internet.
- *Gestión de fallos.* El objetivo de la gestión de fallos es registrar, detectar y contrarrestar las condiciones de fallo de la red. La línea entre la gestión de fallos y la gestión del rendimiento es bastante difusa. Podemos pensar en la gestión de fallos como en la gestión inmediata de fallos transitorios de la red (por ejemplo, fallos hardware o software de un router, un host o de un enlace), mientras que la gestión del rendimiento tiene el objetivo a más largo plazo de proporcionar niveles aceptables de rendimiento en presencia de demandas variables de tráfico y fallos ocasionales de los dispositivos de la red. Al igual que con la gestión del rendimiento, el protocolo SNMP desempeña un papel fundamental en la gestión de fallos.
- *Gestión de configuración.* La gestión de la configuración permite a un administrador de la red controlar qué dispositivos se encuentran dentro de la red administrada y las configuraciones hardware y software de dichos dispositivos. En [RFC 3139] se ofrece una panorámica de la gestión de la configuración y de los requisitos para las redes basadas en IP.

- *Gestión de cuentas.* La gestión de cuentas permite a los gestores de la red especificar, registrar y controlar el acceso de los usuarios y de los dispositivos a los recursos de la red. Las cuotas de uso, la facturación basada en el uso y la asignación de privilegios de acceso a los recursos caen dentro del campo de la gestión de cuentas.
- *Gestión de la seguridad.* El objetivo de la gestión de la seguridad es controlar el acceso a los recursos de red de acuerdo con alguna política bien definida. Los centros de distribución de claves que hemos estudiado en la Sección 8.3 son componentes de la gestión de la seguridad. Otro componente crucial es el uso de cortafuegos para monitorizar y controlar los puntos de acceso externos a la red de la organización, tema que hemos estudiado en la Sección 8.9.

En este capítulo sólo vamos a cubrir los rudimentos de la gestión de red. Nuestro enfoque será conscientemente limitado; examinaremos únicamente la *infraestructura* necesaria para la gestión de red: la arquitectura global, los protocolos de gestión de red y la base de información que permite a un administrador mantener la red en funcionamiento. *No* vamos a cubrir los procesos de toma de decisiones del administrador de red, quien debe planificar, analizar y responder a la información de gestión enviada hacia el NOC. En este área habría que tener en cuenta temas tales como la identificación y gestión de fallos [Katzela 1995; Medhi 1997; Steinder 2002], la detección proactiva de anomalías [Thottan 1998], la correlación de alarmas [Jakobson 1993] y otros. Tampoco vamos a cubrir el tema más amplio de la gestión de servicios [Saydam 1996; RFC 3052; AT&T SLM 2006]: la provisión de recursos tales como el ancho de banda, la capacidad de los servidores y los demás recursos de computación/comunicación necesarios para satisfacer los requisitos de servicio específicos de la misión de una empresa. En esta última área existen estándares tales como TMN [Glitho 1995; Sidor 1998] y TINA [Hamada 1997] que son más completos (y posiblemente más complejos) de cara a tratar estos problemas más generales.

Una cuestión que a menudo se plantea es "¿Qué es la gestión de red?". Las explicaciones que hasta ahora hemos proporcionado demostraban la necesidad de la gestión de red e ilustraban unos cuantos usos de la misma. Vamos a concluir esta sección con una definición de una única frase (aunque un poco larga) del concepto de gestión de red, definición que hemos extraído de [Saydam 1996]:

> *"La gestión de red incluye la implantación, integración y coordinación del hardware, el software y los elementos humanos para monitorizar, probar, sondear, configurar, analizar, evaluar y controlar los recursos de red y los elementos necesarios para satisfacer los requisitos de respuesta en tiempo real, de rendimiento operacional y de calidad de servicio a un coste razonable."*

Parece casi un trabalenguas pero es una definición práctica bastante buena. En las siguientes secciones añadiremos un poco de sustancia a esta definición más bien esquemática de la gestión de red.

9.2 Infraestructura para la gestión de red

Hemos visto en la sección anterior que la gestión de red requiere la capacidad de "monitorizar, probar, sondear, configurar,... y controlar" los componentes hardware y software de una

red. Puesto que los dispositivos de red están distribuidos, esto requerirá como mínimo que el administrador de red sea capaz de recopilar datos (por ejemplo, para propósitos de monitorización) desde una entidad remota y realizar cambios en dicha entidad remota (por ejemplo, para controlarla). Una analogía humana nos resultará bastante útil aquí para entender la infraestructura necesaria para la gestión de red.

Imagine que es usted el director de una gran organización que dispone de sucursales distribuidas por todo el mundo. Su trabajo consiste en asegurarse de que los distintos componentes de su organización operen sin ningún tipo de problema. ¿Cómo lo haría? Como mínimo, recopilará periódicamente datos de sus sucursales mediante informes y diversas medidas cuantitativas de actividad, productividad y presupuesto. Ocasionalmente, pero no siempre, recibirá notificaciones explícitas cuando se produzca un problema en una de las sucursales; el director de sucursal que desee ascender en la escala corporativa (quizá para quitarle a usted el trabajo) puede enviarle informes no solicitados que indiquen lo bien que están marchando las cosas en su sucursal. Usted analizará los informes que reciba confiando en ver que las operaciones se desarrollan sin ningún problema en todas partes, pero encontrándose de vez en cuando, sin ninguna duda, con diversos problemas que requieran su atención. Puede entonces iniciar un diálogo personal con una de sus sucursales problemáticas, recopilando más datos con el fin de comprender el problema y luego emitiendo una orden ejecutiva ("¡Haz este cambio!") al director de la sucursal.

Dentro de este escenario humano bastante común hay implícita una infraestructura de control de la organización: el director (usted), los lugares remotos que se están controlando (las sucursales), los agentes remotos (los directores de sucursal), los protocolos de comunicación (para transmitir los informes y datos estándar y para los diálogos personales) y los datos (el contenido de los informes y las medidas cuantitativas de la actividad, la productividad y los presupuestos). Cada uno de los componentes de este escenario organizativo humano tiene su correspondiente componente en el campo de la gestión de red.

La arquitectura de un sistema de gestión de red es conceptualmente idéntica al de esta sencilla analogía organizativa humana. El campo de la gestión de red tiene su propia terminología específica para los diversos componentes de una arquitectura de gestión de red, por lo que aquí vamos a adoptar dicha terminología. Como se muestra en la Figura 9.2, hay tres componentes principales en una arquitectura de gestión de red: una entidad gestora (el director en nuestra analogía anterior, es decir, usted), los dispositivos gestionados (las sucursales) y un protocolo de gestión de red.

La **entidad gestora** es una aplicación, normalmente con intervención humana, que se ejecuta en una estación central de gestión de red situada en el Centro de operaciones de red (NOC). La entidad gestora es el punto focal de la actividad de administración de la red; controla la recopilación, procesamiento, análisis y/o visualización de la información de gestión de la red. Es aquí donde se inician las acciones para el control del comportamiento de la red y donde el administrador interactúa con los dispositivos que forman la red.

Un **dispositivo gestionado** es un equipo de red (incluyendo su software) que forma parte de una red gestionada. Sería la sucursal dentro de nuestra analogía humana. Un dispositivo gestionado puede ser un host, un router, un puente, un concentrador, una impresora o un módem. Dentro de un dispositivo gestionado pueden existir diversos **objetos gestionados**. Estos objetos gestionados son los propios elementos hardware contenidos en el dispositivo gestionado (por ejemplo, una tarjeta de interfaz de red) y los conjuntos de parámetros de configuración para los distintos elementos hardware y software (por ejemplo, un protocolo de enrutamiento intradominio como RIP). En la analogía humana, los

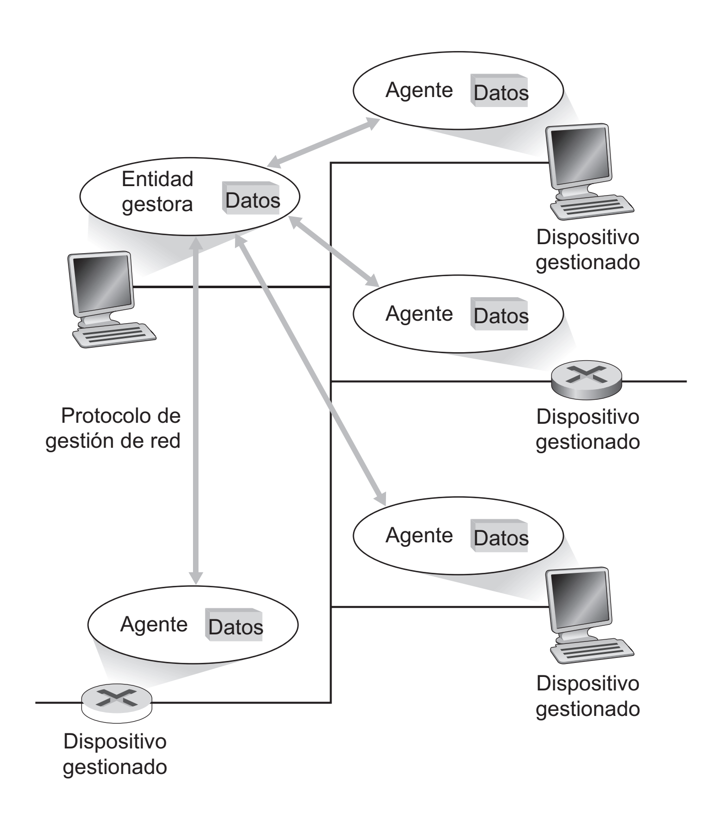


Figura 9.2 • Principales componentes de una arquitectura de gestión de red.

objetos gestionados podrían ser los departamentos de cada sucursal. Estos objetos gestionados tienen asociados distintos elementos de información que se recopilan en una **Base de información de gestión (MIB, *Management Information Base*)**; como veremos, los valores de estos elementos de información están a disposición de la entidad gestora (que en muchos casos también puede modificarlos). En nuestra analogía humana, la MIB se corresponde con los datos cuantitativos (medidas de actividad, productividad y presupuestos, siendo estas últimas configurables por la entidad gestora) intercambiados entre la sucursal y la oficina principal. Estudiaremos en detalle las bases MIB en la Sección 9.3. Finalmente, en cada dispositivo gestionado reside también un **agente de gestión de red**, un proceso que se ejecuta en el dispositivo gestionado y que se comunica con la entidad gestora, llevando a cabo acciones locales en el dispositivo gestionado bajo control de la entidad gestora. En nuestra analogía humana, un agente de gestión de red sería un director de sucursal.

El tercer elemento de una arquitectura de gestión de red es el **protocolo de gestión de red**. El protocolo se ejecuta entre la entidad gestora y los dispositivos gestionados, permitiendo a aquella consultar el estado de los dispositivos gestionados y llevar a cabo acciones de manera indirecta en estos dispositivos a través de sus agentes. Los agentes pueden utilizar el protocolo de gestión de red para informar a la entidad gestora de la ocurrencia de sucesos excepcionales (por ejemplo, fallos de componentes o violación de los umbrales de rendimiento). Es importante recalcar que el protocolo de gestión de red no se encarga él mismo de administrar la red; simplemente proporciona una serie de capacidades que un

administrador puede utilizar para gestionar ("monitorizar, probar, sondear, configurar, analizar, evaluar y controlar ") la red. Se trata de una distinción sutil, pero de gran importancia.

Aunque la infraestructura para la gestión de red es conceptualmente simple, el vocabulario típico de la gestión de redes ("entidad gestora", "dispositivo gestionado", "agente de gestión" y "Base de información de administración") puede a veces resultar confuso. Por ejemplo, en terminología de gestión de red y en nuestro sencillo escenario de monitorización de hosts, los "agentes de gestión" ubicados en los "dispositivos gestionados" son consultados periódicamente por la "entidad gestora" (una idea simple, pero lingüísticamente liosa). Pero tener presente la analogía organizativa humana y sus obvios paralelismos con la gestión de red resultará de gran ayuda para seguir este capítulo.

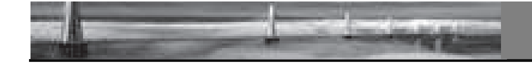
Nuestra exposición acerca de la arquitectura de gestión de red anterior ha sido de carácter genérico y de amplia aplicación a una serie de estándares de gestión de red que han ido siendo propuestos a lo largo de los años. Los estándares de gestión de red comenzaron a madurar a finales de la década de 1980, con el **Elemento común de servicios de información de gestión/Protocolo común de información de gestión** (**CMISE/CMIP,** ***Common Management Information Services Element/Common Management Information Protocol***) de OSI [Piscatello 1993; Stallings 1993; Glitho 1998] y el **Protocolo simple de gestión de red (SNMP,** ***Simple Network Management Protocol*****)** de Internet [RFC 3410; Stallings 1999; Rose 1996] emergiendo como los dos estándares más importantes [Miller 1997; Subramanian 2000]. Ambos se diseñaron para ser independientes de redes y productos específicos de otros fabricantes. Puesto que SNMP fue rápidamente diseñado e implantado cuando se hizo evidente la necesidad de emplear mecanismos de gestión de red, SNMP encontró un amplio uso y aceptación. Actualmente, SNMP es el entorno de gestión de red más ampliamente utilizado e implantado. En la siguiente sección estudiaremos en detalle el protocolo SNMP.

9.3 El entorno de gestión estándar de Internet

Al contrario de lo que podría sugerir el nombre SNMP (*Simple Network Management Protocol*), la gestión de red en Internet es mucho más que un protocolo que permite mover datos de gestión entre una entidad gestora y sus agentes y se ha desarrollado para ser mucho más complejo de lo que el término "simple" puede sugerir. El entorno actual de gestión de Internet estándar tiene sus raíces en el Protocolo simple de monitorización de pasarela (SGMP, *Simple Gateway Monitoring Protocol*) [RFC 1028]. SGMP fue diseñado por un grupo de investigadores universitarios de redes, usuarios y gestores, cuya experiencia con SGMP les permitió diseñar, implementar e implantar SNMP en sólo unos pocos meses [Lynch 1993], algo muy distinto al proceso de estandarización de hoy día. Desde entonces, SNMP ha evolucionado desde SNMPv1 pasando por SNMPv2 hasta la versión más reciente, SNMPv3 [RFC 3410], lanzada en abril de 1999 y actualizada en diciembre de 2002.

Inevitablemente, cuando se describe un entorno para la gestión de redes deben abordarse determinadas cuestiones:

- ¿Qué se quiere decir con monitorizar (desde el punto de vista semántico)? ¿Qué forma de control puede ejercer el administrador de red?
- ¿Cuál es el formato específico de la información comunicada y/o intercambiada?
- ¿Qué protocolo de comunicación se emplea para intercambiar esta información?



PRÁCTICA

CENTRO DE OPERACIONES DE RED DE SPRINTLINK

Las redes pueden tener todo tipo de formas y tamaños. Tanto en las redes domésticas más pequeñas como en el ISP de nivel 1 más grande, es cometido del operador de red garantizar que la red está funcionando sin problemas. Pero, ¿qué ocurre en un centro de operaciones de red (NOC) y cuál es realmente el cometido de un operador de red?

Con una red que se extiende por todo el planeta, Sprint opera una de las redes IP de nivel 1 más grandes del mundo. Conocida como SprintLink (puede obtener más información en www.sprint.com), la red tiene unos 70 puntos de presencia (los POP son posiciones en las que hay routers IP de SprintLink y son los lugares donde los clientes se interconectan con la red) y unos 800 routers. ¡Esto es un montón de ancho de banda! El NOC principal de SprintLink se encuentra en Reston, VA, y los NOC de reserva están en Florida, Georgia y Kansas City. Sprint también mantiene centros NOC para sus redes ATM, frame-relay y de transporte por fibra óptica subyacentes. En cualquier instante, un equipo de cuatro centros de operaciones de red monitoriza y gestiona los equipos que se encuentran en el núcleo de la red IP de SprintLink. También hay otro equipo disponible para tratar cualquier informe de problemas por parte del cliente y para responder a sus solicitudes. La automatización en la monitorización (correlación de alarmas, identificación de fallos y restauración del servicio), la gestión de las configuraciones y la generación de informes de problemas de los clientes, hacen posible que este pequeño grupo de operadores gestionen una red tan grande y compleja.



Técnicos de Sprint monitorizando el estado de la red en centros de operaciones como el de la imagen superior.

(*Continúa*)

Cuando surgen problemas, el principal objetivo del operador de SprintLink es restaurar rápidamente el servicio a los clientes. Los operadores del NOC deciden el orden de las preferencias ante el problema y llevan a cabo los pasos de diagnóstico y de restauración siguiendo un conjunto de procedimientos en respuesta a un conjunto conocido de problemas. Los problemas que no pueden ser diagnosticados de forma inmediata o que no pueden ser resueltos por los operadores dentro de un periodo de tiempo con un nivel de severidad especificado (por ejemplo, 15 minutos), son transferidos al siguiente nivel de soporte. El centro de asistencia técnica nacional (NTAC, *National Technical Assistance Center*) de Sprint proporciona dicho soporte. Los miembros del NTAC son responsables de profundizar en las causas raíz de los problemas; escriben procedimientos operativos para el NOC y trabajan con los proveedores de equipos (por ejemplo, con los fabricantes de routers) para diagnosticar y solucionar problemas relacionados con los equipos, cuando es necesario. Aproximadamente el 90 por ciento de los problemas son tratados directamente por los técnicos e ingenieros del NOC. El personal del NOC y el NTAC interactúa con los demás equipos, incluyendo a los NOC asociados (internos y externos) y a los equipos de operaciones de campo de Sprint que son "los ojos, los oídos y las manos" en los puntos de presencia (POP) de Sprint.

Como hemos mencionado anteriormente en este capítulo, la "gestión de red" dentro de SprintLink (así como en otros ISP) ha evolucionado desde la gestión de fallos, pasando por la gestión del rendimiento hasta la gestión de servicios, poniendo cada vez un mayor énfasis en las necesidades del cliente. Aunque se centran en las necesidades del cliente, los operadores de Sprint también se enorgullecen de la excelencia operacional y de su papel a la hora de mantener y salvaguardar la infraestructura de red global de uno de los ISP más grandes del mundo.

Recordemos nuestra analogía de la organización humana de la sección anterior. El director y los directores de las sucursales deben ponerse de acuerdo acerca de las medidas relativas a la actividad, la productividad y el presupuesto empleadas para informar sobre el estado de una sucursal. De forma similar, necesitarán acordar las acciones que puede tomar el director (por ejemplo, recortar el presupuesto, pedir al director de una sucursal que cambie determinados aspectos del funcionamiento de la sucursal o despedir al personal y cerrar una sucursal). En un nivel de detalle mayor, tendrán que acordar también la forma en que se comunicarán los datos. Por ejemplo, ¿en qué moneda (dólar o euro) se realizan los presupuestos? ¿En qué unidades se mide la productividad? Aunque estos detalles pueden parecer triviales, deben ser acordados. Por último, será necesario especificar la forma en que la información se traslada entre la oficina principal y las sucursales (es decir, el protocolo de comunicaciones).

El entorno de gestión estándar de Internet se ocupa de estas cuestiones. Este entorno consta de cuatro partes:

- Definiciones de los *objetos de gestión de red*, conocidos como objetos MIB. En el entorno de gestión estándar de Internet, la información de gestión se representa como una colección de objetos gestionados que juntos forman un almacén virtual de información, conocido como Base de información de gestión (MIB, *Management Information Base*). Un objeto MIB puede ser un contador, tal como el número de datagramas IP descartados en un router debido a los errores existentes en la cabecera de un datagrama IP o el número de errores de detección de portadora en una tarjeta de interfaz Ethernet; informa-

ción descriptiva, como la versión del software que está ejecutándose en un servidor DNS; información de estado, como por ejemplo si un determinado dispositivo está funcionando correctamente; o información específica del protocolo, como por ejemplo un camino de enrutamiento hasta un destino. Los objetos MIB definen por tanto la información de gestión mantenida por un dispositivo gestionado. Los objetos MIB relacionados se recopilan en **módulos MIB**. En nuestra analogía de la organización humana, la base MIB define la información transmitida entre la oficina principal y las sucursales.

- Un *lenguaje de definición de datos,* conocido como Estructura de la información de gestión (SMI, *Structure of Management Information*). SMI define los tipos de datos, un modelo de objeto y reglas para escribir y revisar la información de gestión. Los objetos MIB se especifican en este lenguaje de definición de datos. En nuestra analogía de la organización humana, la SMI se emplea para definir los detalles del *formato* de la información que va a ser intercambiada.
- Un *protocolo, SNMP.* SNMP se utiliza para transmitir información y comandos entre una entidad gestora y un agente que se ejecuta en un dispositivo de red gestionado, en representación de esa entidad.
- *Capacidades de administración y seguridad.* La adición de estas capacidades representa la principal mejora de SNMPv3 respecto a SNMPv2.

La arquitectura de gestión de red de Internet, por diseño, es por tanto modular, con un lenguaje de definición de datos y una MIB independientes del protocolo, y un protocolo independiente de la MIB. Es interesante reseñar que esta arquitectura modular fue inicialmente utilizada para facilitar la transición de una gestión de red basada en SNMP a un entorno de gestión de red que estaba siendo desarrollado por ISO, la arquitectura de gestión de red competidora cuando se concibió SNMP, aunque fue una transición que nunca llegó a ocurrir. Sin embargo, con el paso del tiempo, la modularidad del diseño de SNMP le ha permitido evolucionar a través de tres revisiones fundamentales, desarrollándose de forma independiente cada una de las cuatro partes principales de SNMP mencionadas más arriba. Evidentemente, se tomó una decisión correcta respecto a la modularidad, ¡aunque por una razón equivocada!

En las siguientes subsecciones vamos a examinar en detalle los cuatro componentes principales del entorno de gestión estándar de Internet.

9.3.1 Estructura de la información de gestión (SMI)

La **Estructura de la información de gestión (SMI, *Structure of Management Information*)** (un componente con un nombre bastante extraño del entorno de gestión de red que no sugiere nada acerca de su funcionalidad) es el lenguaje utilizado para definir la información de gestión que reside en una entidad de red gestionada. Tal lenguaje de definición es necesario para garantizar que la sintaxis y la semántica de los datos de gestión de red están bien definidos y no resultan ambiguos. Observe que la SMI no define una instancia específica de los datos contenidos en una entidad de red gestionada, sino el lenguaje en el que tal información se especifica. Los documentos que describen la SMI para SNMPv3 (que de forma bastante confusa se denomina SMIv2) son [RFC 2578; RFC 2579; RFC 2580]. Examinemos la SMI de abajo hacia arriba, comenzando por los tipos de datos básicos. A continuación, veremos cómo se describen en SMI los objetos gestionados y luego cómo los objetos gestionados relacionados se agrupan en módulos.

Tipos de datos básicos de SMI

El documento RFC 2578 especifica los tipos de datos básicos en el lenguaje de definición de módulos MIB de SMI. Aunque la SMI está basada en el lenguaje de definición de objetos ASN.1 (*Abstract Syntax Notation One*, Notación de sintaxis abstracta uno) [ISO 1987; ISO X.680 2002] (véase la Sección 9.4), se han añadido bastantes tipos de datos específicos de SMI como para considerar que SMI es por sí misma un lenguaje de definición de datos. En la Tabla 9.1 se enumeran los 11 tipos de datos básicos definidos en el documento RFC 2578. Además de estos objetos escalares, también es posible imponer una estructura tabular a una colección ordenada de objetos MIB utilizando la construcción `SEQUENCE OF`; si desea conocer más detalles consulte el documento RFC 2578. La mayor parte de los tipos de datos de la Tabla 9.1 serán familiares (además de que se explican por sí mismos) para la mayoría de los lectores. El único tipo de datos que vamos a ver más detalladamente enseguida es el tipo `OBJECT IDENTIFIER`, que se utiliza para dar nombre a un objeto.

Construcciones de nivel superior de SMI

Además de los tipos de datos básicos, el lenguaje de definición de datos SMI también proporciona construcciones del lenguaje de nivel superior.

La construcción `OBJECT-TYPE` se utiliza para especificar el tipo de datos, el estado y la semántica de un objeto gestionado. Colectivamente, estos objetos gestionados contienen los datos de gestión que constituyen la base de la gestión de la red. Existen más de 10.000 objetos definidos en varios documentos RFC de Internet [RFC 3410]. La construcción

Tipo de datos	Descripción
INTEGER	Entero de 32 bits, como se define en ASN.1, con un valor comprendido entre -2^{31} y $2^{31} - 1$, ambos inclusive, o un valor de una lista de posibles valores constantes con nombre.
Integer32	Entero de 32 bits con un valor comprendido entre -2^{31} y $2^{31} - 1$, ambos inclusive.
Unsigned32	Entero de 32 bits sin signo en el rango de 0 a $2^{32} - 1$, ambos inclusive.
OCTET STRING	Cadena de bytes en formato ASN.1 que representa datos binarios arbitrarios o textuales, de hasta 65.535 bytes de longitud.
OBJECT IDENTIFIER	Nombre estructurado en formato ASN.1 asignado administrativamente; véase la Sección 9.3.2.
IPaddress	Dirección de Internet de 32 bits, en orden de byte de red.
Counter32	Contador de 32 bits que se incrementa de 0 a $2^{32} - 1$ y luego vuelve a 0.
Counter64	Contador de 64 bits.
Gauge32	Entero de 32 bits que no aumenta por encima de $2^{32} - 1$ ni disminuye por debajo de 0 cuando se incrementa o decrementa.
TimeTicks	Tiempo, medido en 1/100 de segundo desde algún suceso.
Opaque	Cadena ASN.1 no interpretada, necesaria por cuestiones de compatibilidad hacia abajo.

Tabla 9.1 • Tipos de datos básicos de la SMI.

OBJECT-TYPE tiene cuatro cláusulas. La cláusula SYNTAX de una definición OBJECT-TYPE especifica el tipo de datos básico asociado con el objeto. La cláusula MAX-ACCESS especifica si el objeto gestionado puede ser leído, escrito, creado o tiene su valor incluido en una notificación. La cláusula STATUS indica si la definición del objeto está actualizada y es válida, si es obsoleta (en cuyo caso no debe implementarse, ya que su definición sólo está incluida con propósitos históricos), o está en desuso (obsoleta pero implementable por cuestiones de interoperabilidad con implementaciones más antiguas). La cláusula DESCRIPTION contiene una definición textual legible del objeto, que "documenta" el propósito del objeto gestionado y debería proporcionar toda la información semántica necesaria para implementar el objeto gestionado.

Como ejemplo de construcción OBJECT-TYPE, considere la definición del tipo de objeto ipSystemStatsInDelivers del [RFC 4293]. Este objeto define un contador de 32 bits que controla el número de datagramas IP que se han recibido en el dispositivo gestionado y que se han entregado con éxito a un protocolo de la capa superior. La última línea de esta definición hace referencia al nombre de este objeto, un tema que abordaremos en la siguiente subsección.

```
ipSystemStatsInDelivers OBJECT-TYPE
    SYNTAX      Counter32
    MAX-ACCESS read-only
    STATUS      current
    DESCRIPTION
          "The total number of datagrams successfully
          delivered to IPuser-protocols (including ICMP).

          When tracking interface statistics, the counter
          of the interface to which these datagrams were
          addressed is incremented. This interface might
          not be the same as theinput interface for
          some of the datagrams.

          Discontinuities in the value of this counter can occur at re-initial-
          ization of the management system, and at other times as indicated by
          the value of ipSystemStatsDiscontinuityTime."
    ::= { ipSystemStatsEntry 18 }
```

La construcción MODULE-IDENTITY permite agrupar objetos relacionados dentro de un "módulo". Por ejemplo, [RFC 4293] especifica el módulo MIB que define los objetos gestionados (incluyendo ipSystemStatsInDelivers) que permiten gestionar implementaciones del Protocolo de Internet (IP) y su Protocolo de mensajes de control de Internet (ICMP) asociado. [RFC 4022] especifica el módulo MIB para TCP y el documento [RFC 4133] especifica el módulo MIB para UDP. [RFC 4502] define el módulo MIB para la monitorización remota RMON. Además de contener las definiciones OBJECT-TYPE de los objetos gestionados dentro del módulo, la construcción MODULE-IDENTITY contiene cláusulas para documentar la información de contacto del autor del módulo, la fecha de la última actualización, un historial de revisiones y una descripción textual del módulo. Considere como ejemplo la siguiente definición de módulo para la gestión del protocolo IP:

```
ipMIB MODULE-IDENTITY
    LAST-UPDATED "200602020000Z"
    ORGANIZATION "IETF IPv6 MIB Revision Team"
    CONTACT-INFO
          "Editor:
          Shawn A. Routhier
          Interworking Labs
          108 Whispering Pines Dr. Suite 235
          Scotts Valley, CA 95066
          USA
          EMail: <sar@iwl.com>"
    DESCRIPTION
          "The MIB module for managing IP and ICMP implementations, but
          excluding their management of IP routes.

          Copyright (C) The Internet Society (2006).
          This version of this MIB module is part of
          RFC 4293; see the RFC itself for full legal notices."

    REVISION       "200602020000Z"
    DESCRIPTION
          "The IP version neutral revision with added IPv6 objects for ND,
          default routers, and router advertisements. As well as being the
          successor to RFC 2011, this MIB is also the successor to RFCs
          2465 and 2466. Published as RFC 4293."

    REVISION       "199411010000Z"
    DESCRIPTION
          "A separate MIB module (IP-MIB) for IP and ICMP management
          objects. Published as RFC 2011."

    REVISION       "199103310000Z"
    DESCRIPTION
          "The initial revision of this MIB module was part of MIB-II,
          which was published as RFC 1213."
    ::= { mib-2 48}
```

La construcción `NOTIFICATION-TYPE` se utiliza para especificar la información relacionada con los mensajes `InformationRequest` y `SNMPv2-Trap` generados por un agente o por una entidad gestora (véase la Sección 9.3.3). Esta información incluye una descripción (`DESCRIPTION`) textual acerca de cuándo deben ser enviados los mensajes, así como una lista de los valores que deben ser incluidos en el mensaje generado; si desea conocer más detalles, consulte el documento [RFC 2578]. La construcción `MODULE-COMPLIANCE` define el conjunto de objetos gestionados en un módulo que un agente debe implementar. La construcción `AGENT-CAPABILITIES` especifica las capacidades de los agentes con respecto a las definiciones de notificación de objetos y sucesos.

9.3.2 Base de información de gestión (MIB)

Como hemos mencionado anteriormente, puede pensarse en la **Base de información de gestión, MIB**, como en un almacén virtual de información, que aloja objetos gestionados cuyos valores reflejan colectivamente el "estado" actual de la red. Estos valores pueden ser consultados y/o definidos por una entidad gestora enviando mensajes SNMP al agente que se está ejecutando en un dispositivo gestionado en representación de la entidad gestora. Los objetos gestionados se especifican utilizando la construcción SMI `OBJECT-TYPE` que hemos explicado anteriormente y se recopilan en **módulos MIB** utilizando la construcción `MODULE-IDENTITY`.

El IETF ha estado ocupado estandarizando los módulos MIB asociados con routers, hosts y otros equipos de red. Esta tarea incluye datos básicos de identificación de un elemento determinado de hardware e información de gestión acerca de los protocolos y las interfaces de red del dispositivo. En 2006 había más de 200 módulos MIB basados en estándares e incluso un número mayor de módulos MIB específicos de fabricantes (privados). Con todos estos estándares, el IETF necesitaba definir una forma de identificar y nombrar tanto los módulos estandarizados como los objetos gestionados específicos contenidos en los módulos. En lugar de partir de cero, el IETF adoptó un marco de identificación (denominación) de objetos estandarizados que ya había utilizado la Organización Internacional de Estandarización (ISO). Como ocurre con muchos organismos de estandarización, ISO tenía "grandes planes" para su entorno de identificación de objetos estandarizados: identificar todos los posibles objetos estandarizados (por ejemplo, formatos de datos, protocolos o elementos de información) en cualquier red, independientemente de la organización de estandarización (como por ejemplo, Internet IETF, ISO, IEEE o ANSI), del fabricante del equipo o del propietario de la red. ¡Un magnífico objetivo! El entorno de identificación de objetos adoptado por ISO es parte del lenguaje de definición de objetos ASN.1 (*Abstract Syntax Notation One*) [ISO X.680 2002] que veremos en la Sección 9.4. Los módulos MIB estandarizados tienen acomodo en este entorno de denominación, como vamos a ver a continuación.

Como se muestra en la Figura 9.3, en el entorno de denominación ISO los objetos se nombran de forma jerárquica. Observe que cada punto de ramificación del árbol tiene un nombre y número (mostrado entre paréntesis); por tanto, cualquier punto del árbol puede ser identificado mediante la secuencia de nombres o números que especifica el camino desde la raíz hasta dicho punto del árbol de identificación. Puede encontrar una utilidad basada en Web divertida, pero incompleta y no oficial, que permite recorrer parte del árbol de identificación de objetos (utilizando información de las ramificaciones aportada por voluntarios) en [Alvestrand 1997] y [France Telecom 2006].

En la parte superior de la jerarquía se encuentran ISO y el Sector de Estandarización de Telecomunicaciones de la Unión Internacional de Telecomunicaciones (ITU-T), las dos organizaciones de estándares principales que se ocupan de ASN.1, así como una rama que une los esfuerzos de estas dos organizaciones. Bajo la rama ISO del árbol podemos ver las entradas correspondientes a todos los estándares ISO (1.0) y a los estándares emitidos por organismos de estandarización de distintos países miembros de ISO (1.2). Aunque en la Figura 9.3 no se muestra, bajo Organismos miembros de ISO (1.2) se encuentra USA (1.2.840), y debajo de éste se encuentran una serie de estándares de IEEE, ANSI y otras empresas específicas, entre las que se incluyen RSA (1.2.840.11359) y Microsoft (1.2.840.113556), y bajo éste están los formatos de archivo de Microsoft (1.2.840.113556.4) para diversos productos de Microsoft, como por ejemplo Word (1.2.840.113556.4.2). Pero ahora nosotros estamos interesados en las

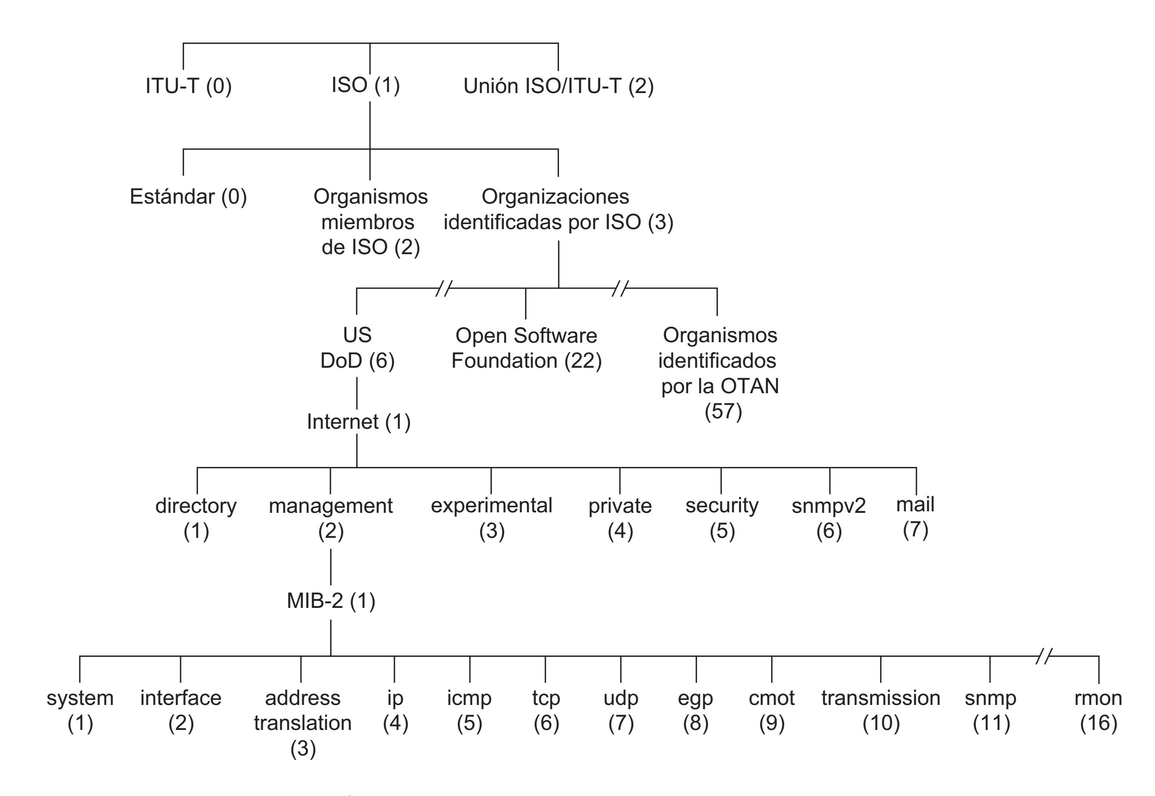


Figura 9.3 • Árbol de identificación de objetos ASN.1.

redes (*no* en los archivos de Microsoft), por lo que vamos a centrar nuestra atención en la rama etiquetada como 1.3, que corresponde a los estándares emitidos por los organismos reconocidos por ISO, entre los que se incluyen el Departamento de Defensa de Estados Unidos, US DoD, (6) (rama bajo la que encontraremos los estándares de Internet), la Open Software Foundation (22), la asociación de líneas aéreas SITA (69), los organismos identificados por la OTAN (57), así como muchas otras organizaciones.

Bajo la rama `Internet` del árbol (1.3.6.1), hay siete categorías. Bajo la rama `private` (1.3.6.1.4) encontramos una lista [IANA 2009b] de los nombres y los códigos empresariales privados de muchos miles de empresas privadas que están registradas en la IANA (*Internet Assigned Numbers Authority*) [IANA 2009a]. Bajo las ramas `management` (1.3.6.1.2) y `MIB-2` (1.3.6.1.2.1) del árbol de identificación de objetos encontramos las definiciones de los módulos MIB estandarizados. ¡Un largo viaje hasta llegar al espacio de nombres ISO!

Módulos MIB estandarizados

El nivel inferior del árbol de la Figura 9.3 muestra algunos de los módulos MIB orientados a hardware más importantes (`system` e `interface`), así como los módulos asociados con algunos de los más importantes protocolos de Internet. [RFC 5000] enumera todos los módulos MIB estandarizados. Aunque los documentos RFC relativos a MIB son bastante tediosos y áridos de leer, resultan instructivos (es decir, es como tener que comer verdura, "es bueno para la salud") para ver unas pocas definiciones de módulos MIB, con el fin de conocer el tipo de información contenida en un módulo.

Identificador del objeto	Nombre	Tipo	Descripción (del RFC 1213)
1.3.6.1.2.1.1.1	`sysDescr`	OCTET STRING	"Nombre completo e identificación de versión del tipo de hardware del sistema, del sistema operativo y del software de red."
1.3.6.1.2.1.1.2	`sysObjectID`	OBJECT IDENTIFIER	ID de objeto asignado por el fabricante que "proporciona un medio fácil y no ambiguo de determinar la clase de máquina que está siendo gestionada."
1.3.6.1.2.1.1.3	`sysUpTime`	TimeTicks	"El tiempo (en centésimas de segundo) desde que la parte de gestión de red del sistema fue reinicializada por última vez."
1.3.6.1.2.1.1.4	`sysContact`	OCTET STRING	"La persona de contacto para este nodo gestionado junto con la información sobre cómo ponerse en contacto con dicha persona."
1.3.6.1.2.1.1.5	`sysName`	OCTET STRING	"Un nombre asignado administrativamente para este nodo gestionado. Por convenio, éste es el nombre de dominio completamente cualificado del nodo."
1.3.6.1.2.1.1.6	`sysLocation`	OCTET STRING	"Localización física de este nodo."
1.3.6.1.2.1.1.7	`sysServices`	Integer32	Un valor codificado que indica el conjunto de servicios disponible en este nodo: físicos (por ejemplo, un repetidor), enlace de datos/subred (por ejemplo, un puente), Internet (por ejemplo, una pasarela IP), terminal a terminal (por ejemplo, un host), aplicaciones.

Tabla 9.2 • Objetos gestionados en el grupo System de MIB-2.

Los objetos gestionados que caen en la rama `system` contienen información general acerca del dispositivo que está siendo gestionado; todos los dispositivos gestionados deben soportar los objetos MIB de `system`. La Tabla 9.2 define los objetos del grupo `system`, tal y como se definen en el documento [RFC 1213]. La Tabla 9.3 define los objetos gestionados en el módulo MIB para el protocolo UDP en una entidad gestora.

9.3.3 Operaciones del protocolo SNMP y correspondencias de transporte

La versión 2 del Protocolo simple de gestión de red (SNMPv2) [RFC 3416] se utiliza para transportar información MIB entre entidades gestoras y agentes que se ejecutan en representación de las entidades gestoras. El uso más común de SNMP es el **modo de solicitud-respuesta** en el que una entidad gestora SNMPv2 envía una solicitud a un agente SNMPv2, el cual la recibe, lleva a cabo ciertas acciones y envía una respuesta a dicha solicitud. Normalmente, una solicitud se utilizará para consultar (recuperar) o modificar (establecer) los valores de objetos MIB asociados con un dispositivo gestionado. Un segundo uso común de SNMP es cuando un agente envía un mensaje no solicitado, conocido como **mensaje TRAP**, a una entidad gestora. Los mensajes TRAP se utilizan para notificar a una entidad gestora una

Identificador del objeto	Nombre	Tipo	Descripción (del RFC 4113)
1.3.6.1.2.1.7.1	`udpInDatagrams`	Counter32	"Número total de datagramas UDP entregados a los usuarios UDP"
1.3.6.1.2.1.7.2	`udpNoPorts`	Counter32	"Número total de datagramas UDP recibidos para los que no había ninguna aplicación en el puerto de destino"
1.3.6.1.2.1.7.3	`udpInErrors`	Counter32	"Número total de datagramas UDP recibidos que no pudieron ser suministrados por razones distintas a que faltara una aplicación en el puerto de destino"
1.3.6.1.2.1.7.4	`udpOutDatagrams`	Counter32	"Numero total de datagramas UDP enviados desde esta entidad"

Tabla 9.3 • Objetos gestionados seleccionados en el módulo UDP de MIB-2.

situación excepcional que ha dado lugar a cambios en los valores de los objetos MIB. Anteriormente en la Sección 9.1 hemos visto que el administrador de red puede desear recibir un mensaje TRAP, por ejemplo, cuando una interfaz deja de funcionar, la congestión alcanza un nivel predefinido en un enlace o tiene lugar algún otro suceso destacable. Observe que existe una serie de compromisos importantes entre el sondeo (interacción solicitud-respuesta) y la captura (*trapping*); véanse los problemas al final del capítulo.

SNMPv2 define siete tipos de mensajes, conocidos genéricamente como Unidades de datos de protocolo (PDU, *Protocol Data Units*), que se enumeran en la Tabla 9.4 y se describen a continuación. El formato de las unidades PDU se muestra en la Figura 9.4.

- Las PDU `GetRequest`, `GetNextRequest` y `GetBulkRequest` son enviadas desde una entidad gestora a un agente para solicitar el valor de uno o más objetos MIB del dispositivo gestionado por el agente. Los identificadores de objeto de los objetos MIB cuyos

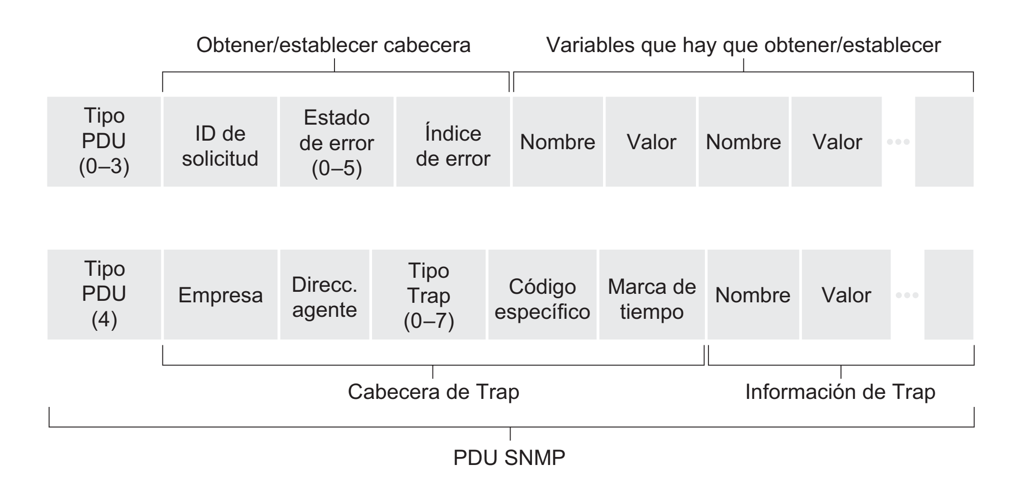


Figura 9.4 • Formato de la PDU de SNMP.

valores están siendo solicitados se especifican en la parte de las variables de la PDU. GetRequest, GetNextRequest y GetBulkRequest difieren en la granularidad de sus solicitudes de datos. GetRequest puede solicitar un conjunto arbitrario de valores MIB; pueden utilizarse múltiples GetNextRequests para recorrer secuencialmente una lista o tabla de objetos MIB; GetBulkRequest permite devolver grandes bloques de datos, evitando la sobrecarga en que se incurre si se enviaron varios mensajes GetRequest o GetNextRequest. En estos tres casos, el agente responde con una PDU Response que contiene los identificadores de objeto y sus valores asociados.

- Una entidad gestora utiliza la PDU SetRequest para establecer el valor de uno o más objetos MIB en un dispositivo gestionado. Un agente responde con una PDU Response con el estado de error "noError" con el fin de confirmar que el valor ha sido definido.
- Una entidad gestora utiliza una PDU InformRequest para notificar a otra entidad gestora cierta información MIB que es remota a la entidad receptora. La entidad receptora responde con una PDU Response con el estado de error "noError" con el fin de confirmar la recepción de la PDU InformRequest.
- El último tipo de PDU de SNMPv2 es el mensaje TRAP. Los mensajes TRAP son generados asíncronamente; es decir, no son generados en respuesta a una solicitud recibida, sino en respuesta a un suceso para el que la entidad gestora requiere una notificación. El documento RFC 3418 define tipos de mensajes TRAP bien conocidos, como el arranque en frío o caliente de un dispositivo, el establecimiento o la pérdida de un enlace, la pérdida de un vecino o un suceso de fallo de autenticación. Una solicitud TRAP recibida no requiere ninguna respuesta por parte de la entidad gestora.

Tipo de PDU de SNMPv2	Emisor-receptor	Descripción
GetRequest	entidad gestora a agente	Obtener el valor de una o más instancias de objeto MIB
GetNextRequest	entidad gestora a agente	Obtener el valor de la siguiente instancia de objeto MIB en una lista o tabla
GetBulkRequest	entidad gestora a agente	Obtener valores en un bloque grande de datos, por ejemplo, en una tabla de gran tamaño
InformRequest	entidad gestora a entidad gestora	Informar a la entidad gestora remota de los valores MIB remotos a su acceso.
SetRequest	entidad gestora a agente	Establecer el valor de una o más instancias de objeto MIB
Response	agente a entidad gestora o entidad gestora a entidad gestora	Generado en respuesta a: GetRequest, GetNextRequest, GetBulkRequest, SetRequest PDU, o InformRequest
SNMPv2-Trap	agente a entidad gestora	Informar al gestor de un suceso excepcional

Tabla 9.4 • Tipos de PDU de SNMPv2.

Dada la naturaleza solicitud-respuesta de SNMPv2, es conveniente señalar aquí que aunque las PDU SNMP pueden ser transportadas por medio de muchos protocolos de transporte distintos, normalmente son transportadas en la carga útil de un datagrama UDP. De hecho, el documento RFC 3417 establece que UDP es "la correspondencia de transporte preferida". Dado que UDP es un protocolo de transporte no fiable, no existe ninguna garantía de que una solicitud, o su respuesta, sea recibida en su destino. La entidad gestora utiliza el campo ID de solicitud de la PDU para numerar las solicitudes que envía a un agente y la respuesta del agente toma su ID de solicitud de la solicitud recibida. Por tanto, el campo ID de solicitud puede ser utilizado por la entidad gestora para detectar las solicitudes y respuestas perdidas. Dependerá de la entidad gestora decidir si retransmite una solicitud si no ha recibido la respuesta correspondiente transcurrido un cierto periodo de tiempo. En particular, el estándar SNMP no obliga a aplicar ningún procedimiento en concreto para las retransmisiones y ni siquiera indica si la retransmisión debe llevarse a cabo. Sólo obliga a que la entidad gestora "actúe de forma responsable respecto a la frecuencia y duración de las retransmisiones". Por supuesto, esto lleva a preguntarse cómo debe actuar un protocolo "responsable".

9.3.4 Seguridad y administración

Los diseñadores de SNMPv3 han dicho que "SNMPv3 puede interpretarse como SNMPv2 con una serie de capacidades adicionales de seguridad y administración" [RFC 3410]. Realmente, existen cambios en SNMPv3 respecto de SNMPv2, pero en ninguna parte estos cambios son más evidentes que en el área de la administración y la seguridad. El papel central de la seguridad en SNMPv3 fue particularmente importante, ya que la falta de una adecuada seguridad hizo que SNMP se usara principalmente para monitorizar más que para cuestiones de control (por ejemplo, rara vez se emplea `SetRequest` en SNMPv1).

Puesto que SNMP ha madurado a lo largo de tres versiones, su funcionalidad ha crecido pero también el número de documentos estándar relacionados con SNMP. Esto lo demuestra el hecho de que existe incluso un documento RFC [RFC 3411] que "describe una arquitectura para describir los entornos de gestión de SNMP". Aunque la idea de una "arquitectura" para "describir un entorno" pueda parecer algo compleja, el objetivo del RFC 3411 es admirable: introducir un lenguaje común para describir la funcionalidad y las acciones tomadas por un agente o una entidad gestora SNMPv3. La arquitectura de una entidad SNMPv3 es clara y un recorrido por dicha arquitectura servirá para afianzar nuestra comprensión de SNMP.

Las denominadas **aplicaciones SNMP** están compuestas por un generador de comandos, un receptor de notificaciones y un dispositivo de reenvío proxy (normalmente, todos ellos se encuentran en una entidad gestora); un contestador de comandos y un emisor de notificaciones (que habitualmente están disponibles en un agente) y la posibilidad de otras aplicaciones. El generador de comandos genera las PDU `GetRequest`, `GetNextRequest`, `GetBulkRequest` y `SetRequest` que hemos examinado en la Sección 9.3.3 y controla las respuestas recibidas a estas PDU. El contestador de comandos se ejecuta en un agente y recibe, procesa y responde (utilizando el mensaje `Response`) a las PDU recibidas `GetRequest`, `GetNextRequest`, `GetBulkRequest` y `SetRequest`. La aplicación que origina las notificaciones en un agente genera las PDU `Trap`; estas PDU son recibidas y procesadas en la aplicación receptora de notificaciones de una entidad gestora. La aplicación de reenvío proxy reenvía las PDU de solicitud, notificación y respuesta.

Una PDU enviada por una aplicación SNMP pasa a continuación a través del "motor" SNMP antes de ser enviada mediante el apropiado protocolo de transporte. La Figura 9.5

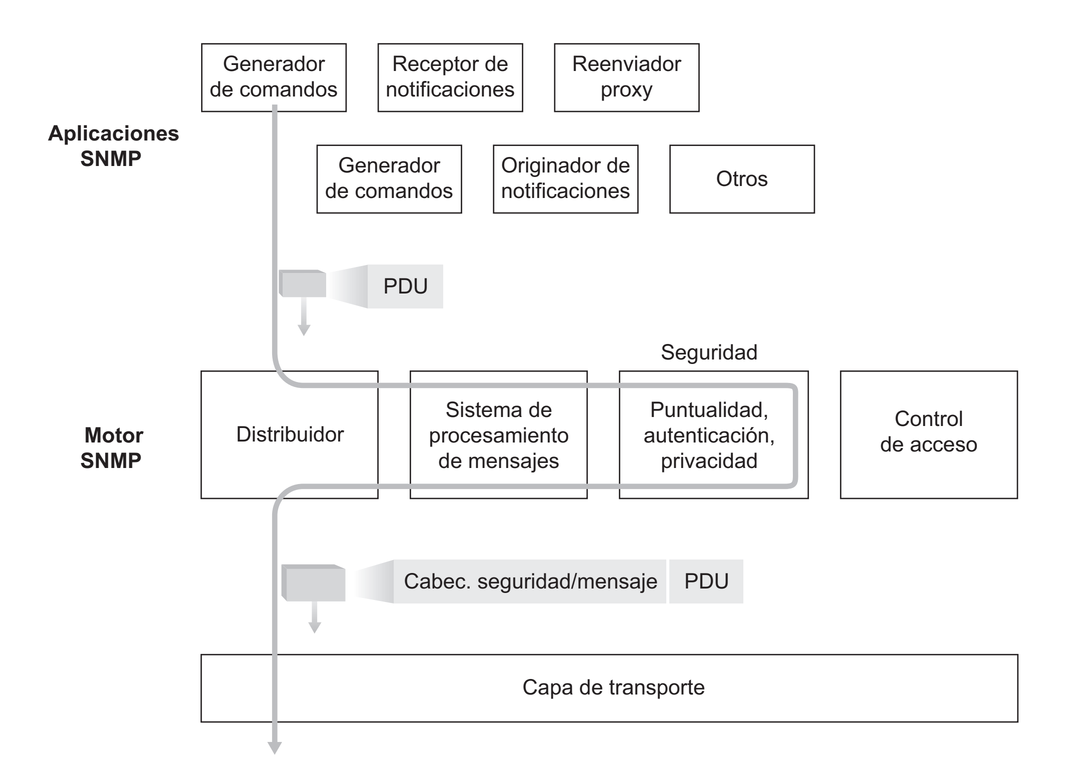


Figura 9.5 • Motor y aplicaciones SNMPv3.

muestra cómo una PDU generada por la aplicación generador de comandos pasa en primer lugar por el módulo de distribución, donde se determina la versión de SNMP. A continuación, la PDU es procesada en el sistema de procesamiento de mensajes, donde se encapsula en una cabecera de mensaje que contiene el número de versión de SNMP, un ID de mensaje y la información de tamaño del mensaje. Si se necesita cifrado o autenticación, también se incluyen los apropiados campos de cabecera que suministran esta información; consulte [RFC 3411] para conocer más detalles. Por último, el mensaje SNMP (la PDU generada por la aplicación más la información de cabecera del mensaje) se pasa al protocolo de transporte apropiado. El protocolo de transporte preferido para los mensajes SNMP es UDP (es decir, los mensajes SNMP son transportados como la carga útil en un datagrama UDP) y el número de puerto preferido para SNMP es el puerto 161. El puerto 162 se utiliza para los mensajes TRAP.

Hemos visto anteriormente que los mensajes SNMP se emplean no sólo para monitorizar, sino también para controlar (por ejemplo, a través del comando `SetRequest`) los elementos de la red. Evidentemente, un intruso que pudiera interceptar los mensajes SNMP y/o generar sus propios paquetes SNMP en la infraestructura de gestión podría afectar a la red. Por tanto, es fundamental que los mensajes SNMP sean transmitidos de forma segura. Sorprendentemente, es sólo en la versión más reciente de SNMP en la que la seguridad ha recibido la atención que merece. La seguridad de SNMPv3 se conoce como **seguridad basada en usuario** [RFC 3414], porque existe el concepto tradicional de usuario, identificado por su nombre de usuario, y una información de seguridad asociada, como por ejemplo una con-

traseña, un valor clave o ciertos privilegios de acceso. SNMPv3 proporciona mecanismos de cifrado, de autenticación, protección contra ataques por reproducción (véase la Sección 8.3) y control de acceso.

- *Cifrado.* Las PDU SNMP pueden cifrarse utilizando el estándar DES (*Data Encryption Standard*) en modo CBC (*Cipher Block Chaining*). Observe que dado que el estándar DES es un sistema de clave compartida, la clave secreta del usuario que cifra los datos tiene que ser conocida por la entidad receptora que tiene que descifrar los datos.
- *Autenticación.* SNMP utiliza la técnica MAC (*Message Authentication Code*) que hemos estudiado en la Sección 8.3.1 para proporcionar tanto autenticación como protección frente a falsificaciones [RFC 2401]. Recuerde que la técnica MAC requiere que tanto el emisor como el receptor tengan una clave secreta común.
- *Protección frente a ataques por reproducción.* Recuerde del Capítulo 8 que pueden emplearse números distintivos para protegerse frente a ataques por reproducción. SNMPv3 adopta una técnica relacionada. Con el fin de asegurarse de que un mensaje recibido no es una reproducción de algún mensaje anterior, el receptor requiere que el emisor incluya un valor en cada mensaje que está basado en un contador del *receptor*. Este contador, que funciona como un número distintivo, refleja el tiempo transcurrido desde que tuvo lugar el último reinicio del software de gestión de red del receptor y el número total de reinicios desde que dicho software del receptor fue configurado por última vez. Siempre y cuando el contador de un mensaje recibido se encuentre dentro de cierto margen de error del valor real del receptor, el mensaje es aceptado como un mensaje que no es una reproducción, momento en el que el mensaje puede ser autenticado y/o descifrado. Para conocer más detalles, consulte [RFC 3414].
- *Control de acceso.* SNMPv3 proporciona un mecanismo de control de acceso basado en vistas [RFC 3415] que controla qué información de gestión de red puede ser consultada y/o definida por qué usuario. Una entidad SNMP mantiene información acerca de las políticas y los derechos de acceso en un Almacén de datos de configuración local (LCD,



PRÁCTICA

Actualmente hay disponibles centenares (si no miles) de productos de gestión de red, todos ellos integrando en cierto grado el entorno de gestión de red y los fundamentos de SNMP que hemos estudiado en esta sección. Un repaso de estos productos queda fuera del ámbito de este texto y (sin duda) lejos de la atención del lector. Por tanto, proporcionamos aquí algunas referencias a algunos de los productos más destacados. Un buen punto de partida para ver una introducción a las herramientas de gestión de red es el Capítulo 12 del libro [Subramanian 2000].

Las herramientas de gestión de red pueden dividirse de forma general entre aquellas que los fabricantes de equipos de red construyen especialmente para la gestión de sus propios equipos y aquellas destinadas a la gestión de red con equipos heterogéneos. Entre las ofertas específicas de los fabricantes se encuentra la serie de herramientas de gestión de red *Network Application Performance Analysis* (NAPA) de Cisco desarrollada para dispositivos Cisco [Cisco NAPA 2009]. Juniper ofrece sistemas de soporte de operación (OSS, *Operation Support Systems*) para el aprovisionamiento de red y soporte SLA/QoS [Juniper 2009].

Local Configuration Datastore). Partes del LCD son accesibles como objetos gestionados que se definen en la MIB de configuración del modelo de control de acceso basado en vistas [RFC 3415], y por tanto pueden ser gestionadas y manipuladas remotamente vía SNMP.

9.4 ASN.1

En este libro hemos abordado una serie de interesantes temas acerca de las redes de computadoras. Sin embargo, esta sección dedicada a ASN.1 es posible que no se encuentre dentro de la lista de los diez temas más interesantes. Como las verduras, el conocimiento de ASN.1 y del amplio tema de los servicios de presentación es algo que es "bueno para ti". ASN.1 es un estándar ISO que se utiliza en una serie de protocolos relacionados con Internet, particularmente en el área de la gestión de red. Por ejemplo, hemos visto en la Sección 9.3 que las variables MIB en SNMP estaban inextricablemente ligadas a ASN.1. Por tanto, aunque el material proporcionado sobre ASN.1 en esta sección puede ser bastante árido, esperamos que el lector considere su importancia.

Con el fin de motivarse, considere el siguiente experimento mental. Suponga que es posible copiar datos de forma fiable directamente desde la memoria de una computadora a la memoria de una computadora remota. Si esto pudiera hacerse, ¿estaría "resuelto" el problema de la comunicación? La respuesta a esta pregunta depende de la definición de "problema de comunicación". Ciertamente, una copia perfecta de memoria a memoria pasaría exactamente los bits y los bytes de un equipo a otro pero, ¿quiere esto decir que cuando el software que se ejecuta en la computadora receptora acceda a estos datos, verá los mismos valores que estaban almacenados en la memoria de la computadora emisora? La respuesta a esta pregunta es "¡no necesariamente!" El quid del problema es que diferentes arquitecturas de computadora, diferentes sistemas operativos y diferentes compiladores emplean convenios distintos para almacenar y representar los datos. Si los datos tienen que transmitirse y almacenarse en distintas computadoras (como ocurre en todas las redes de comunicaciones), entonces está claro que este problema de representación de los datos tiene que solucionarse.

Veamos un ejemplo de este problema. Considere el siguiente fragmento de código C. ¿Cómo se podría disponer esta estructura en memoria?

```
struct {
  char codigo;
  int x;
  } prueba;
prueba.x = 259;
prueba.codigo = 'a';
```

En la parte izquierda de la Figura 9.6 se muestra una posible disposición de estos datos en una arquitectura hipotética: está formada por un único byte de memoria que contiene el carácter `a`, seguido de una palabra de 16 bits que contiene el valor entero 259, almacenado con el byte más significativo en primer lugar. En la parte derecha de la Figura 9.6 se muestra la disposición en memoria de otra computadora. El carácter `a` va seguido por el valor entero almacenado con el byte menos significativo en primer lugar y con el entero de 16 bits alineado al principio de una palabra de 16 bits. Ciertamente, si se realizara una copia literal

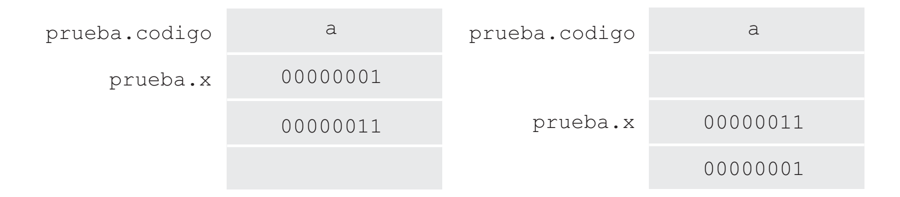


Figura 9.6 • Dos disposiciones de datos distintas en dos arquitecturas diferentes.

entre las memorias de estas computadoras y se empleara la misma definición de estructura para acceder a los valores almacenados, ¡se verían resultados muy diferentes en ambas computadoras!

El hecho de que distintas arquitecturas empleen formatos de datos internos diferentes es un problema real e importante. El problema concreto del almacenamiento de enteros en formatos diferentes es tan común que tiene nombre propio. El sistema *big-endian* para almacenar números enteros coloca en primer lugar los bytes más significativos (en la dirección de almacenamiento menor). El sistema *little-endian* almacena en primer lugar los bytes menos significativos. Los procesadores de Sun SPARC y Motorola emplean el sistema *big-endian*, mientras que los procesa-dores de Intel y DEC/Compaq Alpha emplean el sistema *little-endian*. Los términos *big-endian* y *little-endian* están tomados del libro *Los viajes de Gulliver,* de Jonathan Swift, en el que dos grupos de personas insisten de forma dogmática en hacer una misma cosa de dos formas diferentes (así, la analogía con las arquitecturas de computadora es clara). Un grupo de personas de la tierra de Lilliput insistía en romper los huevos por el extremo más grande ("los *big-endians*"), mientras que el otro grupo quería romperlos por el extremo más pequeño. Esta diferencia fue la causa de una gran rebelión y contienda civil.

Puesto que computadoras distintas almacenan y representan los datos de formas diferentes, ¿cómo deben afrontar esto los protocolos de red? Por ejemplo, si un agente SNMP desea enviar un mensaje `Response` que contiene el valor entero del número de datagramas UDP recibidos, ¿cómo debería representar el valor entero que va enviar a la entidad gestora, utilizando la ordenación *big-endian* o la *little-endian*? Una opción sería que el agente enviara los bytes del entero en el mismo orden en que fueron almacenados en la entidad gestora. Otra opción sería que el agente enviara el entero utilizando su propio orden de almacenamiento y que la entidad receptora reordenara los bytes, en caso necesario. Cualquiera de estas opciones precisa que el emisor y el receptor conozcan el formato de representación de enteros del otro.

Una tercera opción sería disponer de un método independiente de la máquina, independiente del sistema operativo e independiente del lenguaje para describir los números enteros y otros tipos de datos (es decir, disponer de un lenguaje de definición de datos) y de reglas para establecer la forma en que cada uno de los tipos de datos debe ser transmitido a través de la red. Cuando se reciben datos de un determinado tipo, se reciben en un formato conocido y pueden entonces ser almacenados en cualquier formato específico de la máquina requerido. Tanto la SMI que hemos estudiado en la Sección 9.3 como ASN.1 adoptan esta tercera opción. En la jerga de ISO, estos dos estándares describen un **servicio de presentación**: el servicio de transmisión y traducción de información de un formato específico de la

máquina a otro. La Figura 9.7 ilustra un problema de presentación del mundo real; ninguno de los receptores comprende la idea esencial que se está comunicando (que el que habla quiere algo). Como se muestra en la Figura 9.8, un servicio de presentación puede resolver este problema traduciendo la idea a un lenguaje independiente de la persona y de entendimiento común (mediante el servicio de presentación), enviando dicha información al receptor y luego traduciéndola a un lenguaje que el receptor comprenda.

La Tabla 9.5 muestra unos pocos tipos de datos definidos en ASN.1. Recuerde que en nuestro estudio anterior de la SMI hemos hablado de los tipos de datos `INTEGER`, `OCTET STRING` y `OBJECT IDENTIFIER`. Dado que nuestro objetivo aquí (afortunadamente) no es proporcionar una introducción completa a ASN.1, remitimos al lector a los estándares o al libro impreso o en línea [Larmouth 1996] para ver una descripción de los tipos y constructores de ASN.1, tales como `SEQUENCE` y `SET`, que permiten definir tipos de datos estructurados.

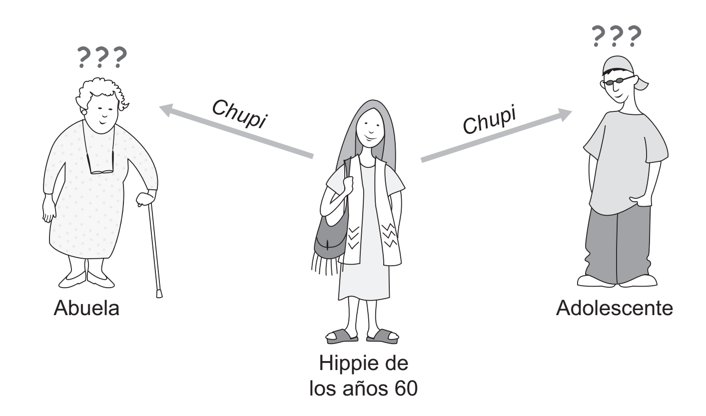


Figura 9.7 • El problema de presentación.

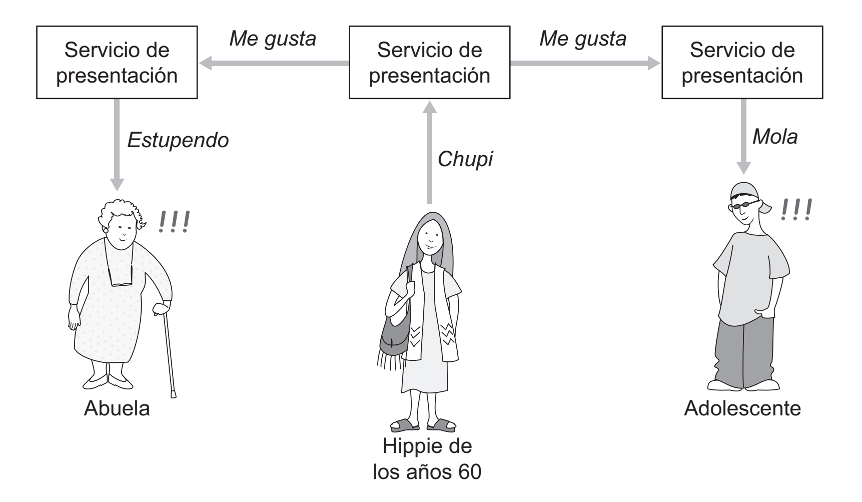


Figura 9.8 • El problema de la presentación resuelto.

Etiqueta	Tipo	Descripción
1	BOOLEAN	El valor es verdadero ("true") o falso ("false")
2	INTEGER	Puede ser arbitrariamente grande
3	BITSTRING	Lista de uno o más bits
4	OCTET STRING	Lista de uno o más bytes
5	NULL	Ningún valor
6	OBJECT IDENTIFIER	Nombre en el árbol de denominación estándar de ASN.1; véase la Sección 9.2.2
9	REAL	Punto flotante

Tabla 9.5 • Tipos de datos ASN.1 seleccionados.

Además de proporcionar un lenguaje de definición de datos, ASN.1 también proporciona **Reglas básicas de codificación (BER, *Basic Encoding Rules*)** que especifican cómo instancias de objetos que han sido definidas utilizando el lenguaje de definición de datos de ASN.1 deben enviarse a través de la red. Las reglas BER adoptan un método denominado **TLV (*Type, Length, Value*; Tipo, Longitud, Valor)** para codificar datos para su transmisión. Para cada elemento de datos que se va a enviar, se transmite el tipo de datos, la longitud del elemento de datos y el valor real de dicho elemento, en este orden. Con este sencillo convenio, los datos recibidos prácticamente se identifican por sí mismos.

En la Figura 9.9 se muestra cómo se enviarían los dos elementos de datos de un ejemplo simple. En este ejemplo, el emisor desea enviar la cadena de caracteres "smith" seguida del valor decimal 259 (que en binario es 00000001 00000011, es decir, un byte de valor 1 seguido de un byte de valor 3) utilizando la ordenación *big-endian*. El primer byte del flujo transmitido tiene el valor 4, lo que indica que el tipo del siguiente elemento de datos es `OCTET STRING`; ésta es la "T" de la codificación TLV. El segundo byte del flujo contiene la longitud de la cadena `OCTET STRING`, en este caso, 5. El tercer byte del flujo da inicio a la cadena `OCTET STRING` de longitud 5 y contiene la representación ASCII de la letra *s*. Los valores T, L y V del siguiente elemento de datos son 2 (el valor de etiqueta de tipo `INTEGER`), 2 (es decir, un entero con una de longitud 2 bytes) y la representación de 2 bytes con ordenación *big-endian* del valor decimal 259.

En nuestra exposición anterior sólo hemos tocado un subconjunto pequeño y simple de ASN.1. Entre los recursos disponibles para aprender más sobre ASN.1 se incluyen el documento de los estándares ASN.1 [ISO X.680 2002], el libro en línea relativo a OSI [Larmouth 1996] y los sitios web relacionados con ASN.1, [OSS 2007] y [France Telecom 2006].

9.5 Conclusión

Terminamos aquí nuestro estudio sobre la gestión de red y todo lo que se refiere a las redes.

En este último capítulo dedicado a la gestión de red, hemos comenzado planteando la necesidad de proporcionar las herramientas apropiadas al administrador de red (la persona

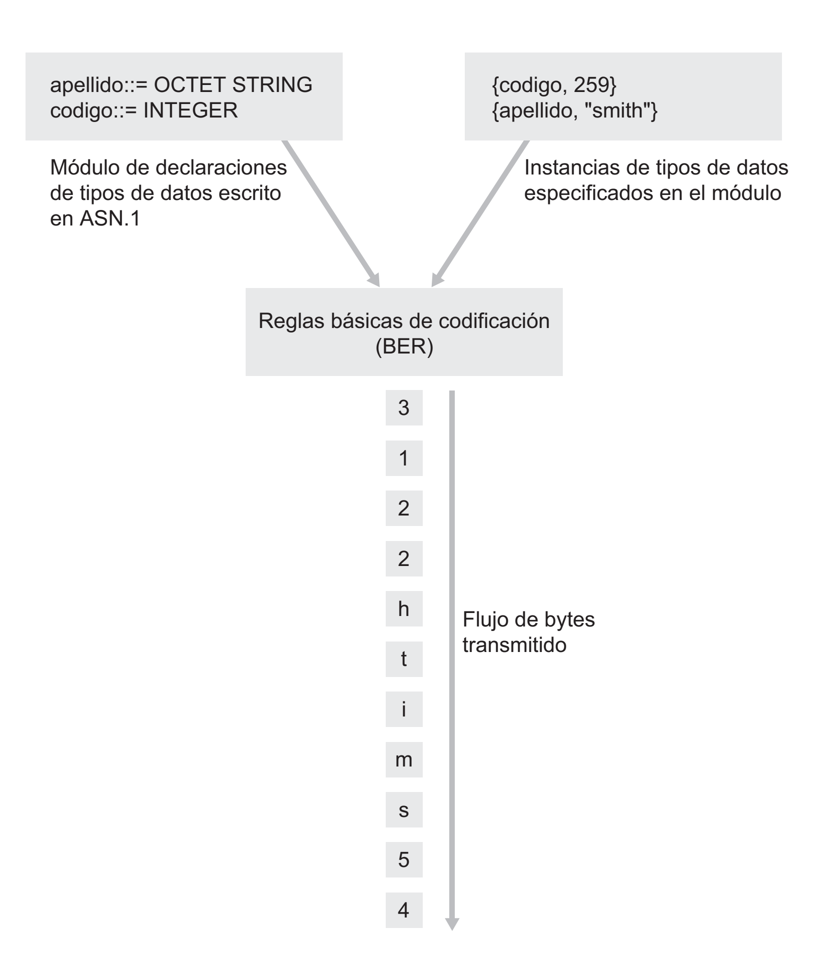


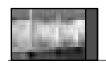
Figura 9.9 • Ejemplo de codificación BER.

cuyo trabajo consiste en mantener la red "activa y funcionando") para monitorizar, probar, sondear, configurar, analizar, evaluar y controlar el funcionamiento de la red. Las analogías presentadas con la gestión de sistemas complejos como las plantas de energía, los aviones y las organizaciones humanas nos han ayudado a entender esta necesidad. Hemos visto que la arquitectura de los sistemas de gestión de red giran en torno a cinco componentes clave: (1) una entidad gestora de red, (2) un conjunto de dispositivos remotos gestionados (desde la entidad gestora), (3) las Bases de la información de gestión (MIB) disponibles en estos dispositivos, que contienen datos acerca de la operación y el estado de los dispositivos, (4) agentes remotos que comunican la información de la MIB y que llevan a cabo acciones bajo el control de la entidad gestora de red y (5) un protocolo para la comunicación entre la entidad gestora y los dispositivos remotos.

A continuación hemos profundizado en los detalles del entorno de gestión estándar de Internet y en el protocolo SNMP en particular. Hemos visto cómo SNMP instancia los cinco componentes clave de una arquitectura de gestión de red y hemos dedicado un tiempo considerable a examinar los objetos MIB, la SMI (el lenguaje de definición de datos para especificar las bases MIB) y el propio protocolo SNMP. Dado que SMI y ASN.1 están estre-

chamente ligados y que ASN.1 desempeña un papel clave en la capa de presentación del modelo de referencia de siete capas ISO/OSI, hemos examinado brevemente ASN.1. Quizá más importante que los detalles del propio ASN.1 fue la necesidad de proporcionar la traducción entre los formatos de datos específicos de las máquinas de una red. Aunque algunas arquitecturas de red reconocen explícitamente la importancia de este servicio disponiendo de una capa de presentación, esta capa está ausente en la pila de protocolos de Internet.

También merece la pena destacar que hay muchos temas sobre la gestión de red que hemos decidido *no* abordar, temas tales como la identificación y gestión de fallos, la detección proactiva de anomalías, la correlación de alarmas y las muchas cuestiones sobre la gestión de servicios (por ejemplo, en oposición a la gestión de red). Aunque importantes, estos temas merecen su propio libro, por lo que el lector puede recurrir a las referencias dadas en la Sección 9.1.



Problemas y cuestiones de repaso

Capítulo 9 Problemas de repaso

SECCIÓN 9.1

R1. ¿Por qué se beneficia un administrador de red de disponer de herramientas de gestión de red? Describa cinco escenarios.

R2. ¿Cuáles son las cinco áreas de gestión de red definidas por la organización ISO?

R3. ¿Cuál es la diferencia entre gestión de red y gestión de servicios?

SECCIÓN 9.2

R4. Defina los siguientes términos: entidad gestora, dispositivo gestionado, agente de gestión, MIB, protocolo de gestión de red.

SECCIÓN 9.3

R5. ¿Cuál es el papel de la SMI en la gestión de red?

R6. ¿Cuál es una diferencia importante entre un mensaje solicitud-respuesta y un mensaje TRAP en SNMP?

R7. ¿Cuáles son los siete tipos de mensajes utilizados en SNMP?

R8. ¿Qué es un “motor SNMP”?

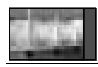
SECCIÓN 9.4

R9. ¿Cuál es el propósito del árbol de identificación de objetos de ASN.1?

R10. ¿Cuál es el papel de ASN.1 en la capa de presentación del modelo de referencia ISO/OSI?

R11. ¿Tiene Internet una capa de presentación? Si no la tiene, ¿cómo se tratan las diferencias entre las arquitecturas de máquinas, por ejemplo, la distinta representación de los enteros en las distintas máquinas?

R12. ¿Cuál es el significado de la codificación TLV?



Problemas

P1. Considere las dos formas en que tiene lugar la comunicación entre una entidad gestora y un dispositivo gestionado: el modo solicitud-respuesta y TRAP. ¿Cuáles son las ventajas y los inconvenientes de estos dos métodos, en términos de (1) costes, (2) tiempo de notificación cuando se producen sucesos excepcionales y (3) robustez con respecto a los mensajes perdidos entre la entidad gestora y el dispositivo?

P2. En la Sección 9.3 hemos visto que era preferible transportar mensajes SNMP en datagramas UDP no fiables. ¿Por qué cree que los diseñadores de SNMP eligieron UDP en lugar de TCP como protocolo de transporte para SNMP?

P3. ¿Qué es el identificador de objeto ASN.1 para el protocolo RMON (véase la Figura 9.3)?

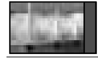
P4. Suponga que trabaja en una empresa de Estados Unidos que desea desarrollar su propia MIB para gestionar una línea de productos. ¿Dónde se registraría dentro del árbol de identificación de objetos (Figura 9.3)? (*Sugerencia:* tendrá que investigar en los RFC u otros documentos para poder responder a esta pregunta.)

P5. Recuerde de la Sección 9.3.2 que una empresa privada puede crear sus propias variables MIB bajo la rama `private` 1.3.6.4. Suponga que IBM deseara crear una MIB para el software de su servidor web. ¿Cuál sería el siguiente cualificador OID después de 1.3.6.1.4? (Para responder a esta pregunta, tendrá que consultar [IANA 2009b]). Realice una búsqueda en la Web y vea si puede encontrar si existe tal MIB en un servidor de IBM.

P6. ¿Por qué cree que la longitud precede al valor en una codificación TLV (en lugar de que ir a continuación del valor)?

P7. Considere la Figura 9.9. ¿Cuál sería la codificación BER de `{codigo, 276} {apellido, “Marco”}`?

P8. Considere la Figura 9.9. ¿Cuál sería la codificación BER de `{codigo, 160} {apellido, “Dario”}`?



Preguntas para la discusión

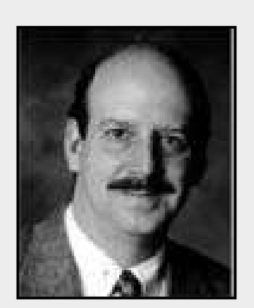
D1. Además de la planta de energía eléctrica o la cabina de un avión, indique una analogía de un sistema distribuido complejo que necesite ser controlado.

D2. Considere el escenario de la Sección 9.1. ¿Qué otras actividades cree que un administrador de red puede desear monitorizar? ¿Por qué?

D3. Lea el documento RFC 789. ¿Cómo podría haberse evitado la caída de ARPAnet de 1980 (o cómo podría haberse simplificado su recuperación) si los administradores de ARPAnet hubieran dispuesto de las herramientas de gestión de red actuales?

UNA ENTREVISTA CON...

Jeff Case



Jeff Case es el fundador y director técnico de SNMP Research, Inc. SNMP (*Simple Network Management Protocol*) es un productor líder de elaboración de estándares de Internet y productos basados en estándares para la gestión de redes. Jeff se graduó en Educación Industrial y Tecnología de Ingeniería Eléctrica y obtuvo los masters en Educación Industrial y en Ingeniería Eléctrica en la Universidad de Purdue. Se doctoró en Educación Técnica en la Universidad de Illinois, Urbana-Champaign.

¿Por qué decidió especializarse en el campo de las redes ?

Siempre me ha fascinado el construir todo tipo de cosas, desde que me dediqué a introducir horquillas del pelo en los enchufes cuando era un niño. Poco a poco fui desarrollando un interés en los equipos de sonido durante mi adolescencia, fabricando sistemas de sonido para grupos de rock que parecían lo suficientemente potentes como para pulverizar el hormigón. Mientras estaba en la universidad, trabajaba reparando televisiones y equipos de radio (principalmente equipos de sonido) y me sentí interesado por las computadoras y por todo aquello que fuera digital, incluyendo tanto el hardware como el software. Comencé a tratar de interconectar todo tipo de equipos extraños. Primero, me dediqué a conectar periféricos con procesadores. Después unos sistemas con otros. El campo de la redes es la más evolucionada de las interfaces y, actualmente, Internet es la más evolucionada de las redes.

¿En qué consistió su primer trabajo en la industria de las computadoras? ¿A qué se dedicaba?

La mayor parte de mis primeros años de profesión la pasé en la universidad de Purdue. A lo largo del tiempo me tocó impartir formación en todos los cursos que definen el currículum de los estudiantes de primer ciclo en Ingeniería Informática y Eléctrica. Mi trabajo incluía diseñar nuevos cursos dedicados a los temas entonces nacientes del hardware y el software de microprocesadores. Por ejemplo, durante un semestre, nuestra clase tenía que diseñar una computadora a partir de los chips componentes construyéndola durante las prácticas de laboratorio del curso; uno de los equipos se encargaba de la unidad central de proceso, otro del subsistema de memoria y otro del subsistema de E/S, etc. Al semestre siguiente, escribíamos el software del sistema para el hardware que habíamos construido.

Fue más o menos al tiempo que comencé a trabajar en tareas de dirección en los sistemas informáticos del campus, encargándome de informar al rector como director de servicios de usuario de los sistemas de computadoras académicos.

¿Qué parte de su trabajo es la que representa un mayor desafío?

Seguir estando al día de todos los cambios, tanto en el campo técnico como en el de los negocios. Soy un director con un perfil bastante técnico y cada vez me es más difícil estar actualizado en lo que respecta a los avances técnicos experimentados en nuestro sector. Mi papel también me exige seguir los cambios que se producen en el entorno empresarial, como por ejemplo en las fusiones y adquisiciones.

¿Cuál cree qué es el futuro de las redes y de Internet?

Un futuro de crecimiento sostenido. Una velocidad cada vez mayor. Una ubicuidad creciente. Un contenido en expansión. Una mayor tensión entre anarquía y dirección centralizada. Más correo basura. Más herramientas para combatir el correo basura. Más problemas de seguridad. Más soluciones de seguridad. Finalmente, debemos estar preparados para lo completamente inesperado.

¿Qué personas le han inspirado profesionalmente?

Mi difunto padre, que fue un hombre de negocios de éxito; Dilbert; los doctores Vint Cerf, Jon Postel, Marshall Rose y Chuck Davin, que son personas bien conocidas en el sector de Internet; Bill Seifert, ahora socio de capital riesgo; el doctor Rupert Evans, mi profesor de tésis; mi esposa, que trabaja conmigo en la empresa y, en último lugar, pero no por ello menos importante, Jesús.

He oído que tiene usted una notable colección de "citas". Cuando trabajaba como profesor, ¿solía ilustrar sus enseñanzas con esas citas?

"Un ejemplo es más valioso que dos libros" (creo que es de Gauss).

"En ocasiones, existe una distancia entre la teoría y la práctica. La distancia entre la teoría y la práctica en teoría no es tan grande como la distancia entre la teoría y la práctica en la práctica." (No tengo ni idea de quién es esta cita.)

¿Cuáles han sido los mayores obstáculos a la hora de desarrollar los estándares Internet?

El dinero. Los factores políticos. El ego. Los errores de liderazgo.

¿Cuál ha sido la utilización más sorprendente de la tecnología SNMP?

Todas ellas. Yo llegué a involucrarme en las cuestiones de gestión de Internet con el único objeto de satisfacer mis propias necesidades de supervivencia. Necesitaba disponer de algunas buenas herramientas para gestionar la infraestructura de red de mi organización. El gran éxito debido a la existencia de muchas otras personas que necesitaban resolver problemas similares es atribuíble a la intuición, a la buena suerte y al trabajo duro. Lo importante es que conseguimos dar muy pronto con la arquitectura correcta.

Referencias

Aclaración sobre los URL: en las referencias que se proporcionan a continuación hemos incluido los URL a páginas web, documentos exclusivamente web y otros materiales que no hayan sido publicados en una conferencia o revista (siempre que hemos podido localizar un URL para dicho material). A diferencia de las ediciones anteriores de este libro, no hemos incluido los URL correspondientes a conferencias y publicaciones en revistas, ya que dichos documentos normalmente pueden encontrarse utilizando el motor de búsqueda disponible en el sitio web de una conferencia (por ejemplo, los documentos de todas las conferencias y workshops de *ACM Sigcomm* están disponibles en http://www.acm.org/sigcomm), o a través de una suscripción digital. Todos los URL que se proporcionan a continuación se validaron (y probaron) en enero de 2009, por lo que es posible que actualmente alguno esté desactualizado. Por favor consulte la versión en línea de este libro (http://www.awl.com/kurose-ross) para obtener la bibliografía actualizada.

Aclaración sobre los documentos RFC (*Request for Comments*) de Internet: hay disponibles copias de los documentos RFC de Internet en muchos sitios. El editor de documentos RFC de la Internet Society (el organismo que supervisa los RFC) mantiene el sitio http://www.rfc-editor.org. En este sitio es posible buscar un RFC específico por título, número o autores, y muestra las actualizaciones de los RFC localizados. Los RFC de Internet pueden estar actualizados o haber quedado obsoletos debido a la existencia de documentos posteriores. Nuestro sitio favorito para consultar los RFC es el original: http://www.rfc-editor.org.

[3Com Addressing 2009] 3Com Corp., "White paper: Understanding IP addressing: Everything you ever wanted to know", http://www.3com.com/other/pdfs/infra/corpinfo/en_US/501302.pdf

[3GPP 2009] Página principal de Third Generation Partnership Project: http://www.3gpp.org/

[802.11 Security 2009] The Unofficial 802.11 Security Web Page, http://www.drizzle.com/~aboba/IEEE/

[Abitz 1993] P. Albitz y C. Liu, *DNS and BIND,* O'Reilly & Associates, Petaluma, CA, 1993.

[Abramson 1970] N. Abramson, "The Aloha System—Another Alternative for Computer Communications", *Proc. 1970 Fall Joint Computer Conference, AFIPS Conference,* pág. 37, 1970.

[Abramson 1985] N. Abramson, "Development of the Alohanet", *IEEE Transactions on Information Theory,* Vol. IT-31, Nº 3 (marzo 1985), págs. 119–123.

[Ahn 1995] J. S. Ahn, P. B. Danzig, Z. Liu y Y. Yan, "Experience with TCP Vegas: Emulation and Experiment", *Proc. 1995 ACM SIGCOMM* (Boston, MA, agosto 1995), págs. 185–195.

[Akamai 2009] Página principal de Akamai: http://www.akamai.com.

[Akella 2003] A. Akella, S. Seshan, A. Shaikh; "An empirical Evaluation of Wide-area Internet Bottlenecks", *Proc. 2003 ACM Internet Measurement Conf.* (Miami FL, noviembre 2003).

[Alvestrand 1997] H. Alvestrand, "Object Identifier Registry", http://www.alvestrand.no/objectid/

[Anderson 1995] J. B. Andersen, T. S. Rappaport, S. Yoshida, "Propagation Measurements and Models for Wireless Communications Channels", *IEEE Communications Magazine* (enero 1995), págs. 42–49.

[Aperjis 2008] C. Aperjis, M.J. Freedman, R. Johari, "Peer-assisted Content Sistribution with Prices", *Proc. ACM CoNEXT'08,* (Madrid, diciembre 2008).

[Appenzeller 2004] G. Appenzeller, I. Keslassy, N. McKeown, "Sizing Router Buffers*", Proc. 2004 ACM SIGCOMM* (Portland, OR, agosto 2004).

[ARIN 1996] ARIN, "IP allocation report", ftp://rs.arin.net/netinfo/ip_network_allocations

[Ash 1998] G. R. Ash, *Dynamic Routing in Telecommunications Networks,* McGraw Hill, NY, NY, 1998.

[ASO-ICANN 2009] The Address Supporting Organization home page, http://www.aso .icann.org

[AT&T SLM 2008] AT&T Business, "AT&T Enterprise Hosting Services Service Guide", http://www.att.com/abs/serviceguide/docs/eh_sg.pdf

[Atheros 2009] Atheros Communications Inc. "Atheros AR5006 WLAN Chipset Product Bulletins", http://www.atheros.com/pt/AR5006Bulletins.htm

[Ayanoglu 1995] E. Ayanoglu, S. Paul, T. F. La Porta, K. K. Sabnani, R. D. Gitlin, "AIRMAIL: A Link-Layer Protocol for Wireless Networks", *ACM ACM/Baltzer Wireless Networks Journal*, 1: 47–60, febrero 1995.

[Bakre 1995] A. Bakre, B. R. Badrinath, "I-TCP: Indirect TCP for Mobile Hosts", *Proc. 1995 Int. Conf. on Distributed Computing Systems (ICDCS)*, mayo 1995, págs. 136–143.

[Balakrishnan 1997] H. Balakrishnan, V. Padmanabhan, S. Seshan, R. Katz, "A Comparison of Mechanisms for Improving TCP Performance Over Wireless Links", *IEEE/ACM Transactions on Networking* Vol. 5, Nº 6 (diciembre 1997),

[Baran 1964] P. Baran, "On Distributed Communication Networks", *IEEE Transactions on Communication Systems,* marzo 1964. Rand Corporation Technical report with the same title (Memorandum RM-3420-PR, 1964). http://www.rand.org/publications/RM/RM3420/

[Bardwell 2004] J. Bardwell, "You Believe You Understand What You Think I Said . . . The Truth About 802.11 Signal And Noise Metrics: A Discussion Clarifying Often-Misused 802.11 WLAN Terminologies", http://www.connect802.com/download/techpubs/2004/you_believe_D100201.pdf

[Baset 2006] S. A. Basset and H. Schulzrinne, "An analysis of the Skype peer-to-peer Internet Telephony Protocol", *Proc. 2006 IEEE Infocom* (Barcelona, España, abril 2006).

[BBC 2001] BBC news online "A Small Slice of Design", abril 2001, http://news.bbc.co.uk/2/hi/science/nature/1264205.stm

[BBC Multicast 2009] BB, "BBC Multicast Trial" http://support.bbc.co.uk/multicast/

[Beheshti 2008] N. Beheshti, Y. Ganjali, M. Ghobadi, N. McKeown, G. Salmon, "Experimental Study of Router Buffer Sizing", *Proc. ACM Internet Measurement Conference, October 2008,* Vouliagmeni, Greece.

[Bender 2000] P. Bender, P. Black, M. Grob, R. Padovani, N. Sindhushayana, A. Viterbi, "CDMA/HDR: A bandwidth-efficient high-speed wireless data service for nomadic users", *IEEE Commun. Mag.,* Vol. 38, Nº 7 (julio 2000) págs. 70–77.

[Berners-Lee 1989] T. Berners-Lee, CERN, "Information Management: A Proposal", marzo 1989, mayo 1990. http://www.w3.org/History/1989/proposal.html

[Berners-Lee 1994] T. Berners-Lee, R. Cailliau, A. Luotonen, H. Frystyk Nielsen, A. Secret, "The World-Wide Web", *Communications of the ACM,* Vol. 37, Nº 8 (agosto 1994), págs. 76–82.

[Bernstein 2009] D. Bernstein, "SYN Cookies", http://cr.yp.to/syncookies.html

[Bertsekas 1991] D. Bertsekas, R. Gallagher, *Data Networks, 2ª ed.,* Prentice Hall, Englewood Cliffs, NJ, 1991.

[Bhimani 1996] A. Bhimani: "Securing the Commercial Internet", *Communications of the ACM,* Vol. 39 Nº 6 (marzo 1996), págs. 29–35.

[Biddle 2003] P. Biddle, P. England, M. Peinado, B. Willman, "The Darknet and the Future of Content Distribution", *2002 ACM Workshop on Digital Rights Management* (noviembre 2002, Washington, D.C.) http://crypto.stanford.edu/DRM2002/darknet5.doc

[Biersack 1992] E. W. Biersack, "Performance evaluation of forward error correction in ATM networks", *Proc. 1999 ACM SIGCOMM* (Baltimore, MD, agosto 1992), págs. 248–257.

[BIND 2009] Internet Software Consortium page on BIND, http://www.isc.org/bind.html

[Bisdikian 2001] C. Bisdikian, "An Overview of the Bluetooth Wireless Technology", *IEEE Communications Magazine*, Nº 12 (diciembre 2001), págs. 86–94.

[Bishop 2003] M. Bishop, *Computer Security: Art and Science,* Boston: Addison Wesley, Boston MA, 2003.

[BitTorrent 2009] Página principal de BitTorrent.org: http://www.bittorrent.org

[Black 1995] U. Black, *ATM Volume I: Foundation for Broadband Networks*, Prentice Hall, 1995.

[Blumenthal 2001] M. Blumenthal, D. Clark, "Rethinking the Design of the Internet: the End-to-end Arguments vs. the Brave New World", *ACM Transactions on Internet Technology,* Vol. 1, Nº 1 (agosto 2001) págs. 70–109.

[Bochman 1984] G. V. Bochmann, C. A. Sunshine, "Formal methods in communication protocol design," *IEEE Transactions on Communications,* Vol. 28, Nº 4 (abril 1980), págs. 624–631.

[Bolot 1994] J-C. Bolot, T. Turletti, "A rate control scheme for packet video in the Internet," *Proc. 1994 IEEE Infocom,* págs. 1216–1223.

[Bolot 1996] J-C. Bolot, A. Vega-Garcia, "Control Mechanisms for Packet Audio in the Internet", *Proc. 1996 IEEE Infocom,* págs. 232–239.

[Bradner 1996] S. Bradner, A. Mankin, *IPng: Internet Protocol Next Generation,* Addison-Wesley, Reading, MA, 1996.

[Brakmo 1995] L. Brakmo, L. Peterson, "TCP Vegas: End to End Congestion Avoidance on a Global Internet", *IEEE Journal of Selected Areas in Communications,* Vol. 13, Nº 8, págs. 1465–1480, octubre 1995.

[Breslau 2000] L. Breslau, E. Knightly, S. Shenker, I. Stoica, H. Zhang, "Endpoint Admission Control: Architectural Issues and Performance", *Proc. 2000 ACM SIGCOMM* (Estocolmo, Suecia, agosto 2000).

[Brodnik 1997] A. Brodnik, S. Carlsson, M. Degemark, S. Pink, "Small Forwarding Tables for Fast Routing Lookups", *Proc. 1997 ACM SIGCOMM* (Cannes, Francia, octubre 1997), págs. 3–15.

[Bush 1945] V. Bush, "As We May Think", *The Atlantic Monthly,* julio 1945. http://www.theatlantic.com/unbound/flashbks/computer/bushf.htm.

[Byers 1998] J. Byers, M. Luby, M. Mitzenmacher, A Rege, "A digital fountain approach to reliable distribution of bulk data", *Proc. 1998 ACM SIGCOMM* (Vancouver, Canadá, agosto 1998), págs. 56–67.

[Cablelabs 2009] Página principal de CableLabs: http://www.cablelabs.com

[CacheLogic 2009] Página principal de CacheLogic: http://www.cachelogic.com

[Caesar 2005] M. Caesar, J Rexford, "BGP Routing Policies in ISP Networks", *IEEE Network Magazine,* vol. 19, Nº 6 (noviembre 2005).

[Caldwell 2009] C. Caldwell, "The Prime Pages", http://www.utm.edu/research/primes/prove

[Cardwell 2000] N. Cardwell, S. Savage, T. Anderson, "Modeling TCP Latency", *Proc. 2000 IEEE Infocom*, (Tel-Aviv, Israel, marzo 2000).

[CASA 2009] Center for Collaborative Adaptive Sensing of the Atmosphere, http://www .casa.umass.edu

[Casado 2007] M. Casado, M. Freedman, J. Pettit, J. Luo, N. McKeown, S. Shenker, "Ethane: Taking Control of the Enterprise", *Proc.* ACM SI*GCOMM '07,* (agosto 2007, Kyoto, Japón).

[Casner 1992] S. Casner, S. Deering, "First IETF Internet Audiocast", *ACM SIGCOMM Computer Communications Review,* Vol. 22, Nº 3 (julio 1992), págs. 92–97.

[Ceiva 2009] Página principal de Ceiva: http://www.ceiva.com/

[CENS 2009] Center for Embedded Network Sensing, http://www.cens.ucla.edu/

[Cerf 1974] V. Cerf y R. Kahn, "A Protocol for Packet Network Interconnection", *IEEE Transactions on Communications Technology,* Vol. COM-22, Nº 5, págs. 627–641.

[CERT 2001–09] CERT, "Advisory 2001–09: Statistical Weaknesses in TCP/IP Initial Sequence Numbers", http://www.cert.org/advisories/CA-2001-09.html

[CERT 2003–04] CERT, "CERT Advisory CA-2003-04 MS-SQL Server Worm", http:// www.cert.org/advisories/CA-2003-04.html

[CERT 2009] Centro de coordinación del CERT: http://www.cert.org/advisories

[CERT Filtering 2009] CERT, "Packet Filtering for Firewall Systems", http://www.cert.org/tech_tips/packet_filtering.html

[Cert SYN 1996] CERT, "Advisory CA-96.21: TCP SYN Flooding and IP Spoofing Attacks", http://www.cert.org/advisories/CA-1998-01.html

[Chao 2001] H. J. Chao, C. Lam, E. Oki, Broadband Packet Switching Technologies—A Practical Guide to ATM Switches and IP Routers, John Wiley & Sons, 2001.

[Chen 2000] G. Chen, D. Kotz, "A Survey of Context-Aware Mobile Computing Research", *Technical Report TR2000-381*, Dept. of Computer Science, Dartmouth College, noviembre 2000. http://www.cs.dartmouth.edu/reports/TR2000-381.pdf

[Chen 2006] K.-T. Chen, C.-Y. Huang, P. Huang, C.-L. Lei, "Quantifying Skype User Satisfaction", *Proc. 2006 ACM SIGCOMM* (Pisa, Italia, septiembre 2006).

[Cheswick 2000] B. Cheswick, H. Burch, S. Branigan, "Mapping and Visualizing the Internet", *Proc. 2000 Usenix Conference* (junio 2000, San Diego).

[Chiu 1989] D. Chiu, R. Jain, "Analysis of the Increase and Decrease Algorithms for Congestion Avoidance in Computer Networks", *Computer Networks and ISDN Systems,* Vol. 17, Nº 1, págs. 1–14. http://www.cs.wustl.edu/~jain/papers/cong_av.htm

[Christiansen 2001] M. Christiansen, K. Jeffay, D. Ott, F. D. Smith, "Tuning Red for Web Traffic", *IEEE/ACM Transactions on Networking*, Vol. 9, Nº 3 (junio 2001), págs. 249–264.

[Chuang 2005] S. Chuang, S. Iyer, N. McKeown, "Practical Algorithms for Performance Guarantees in Buffered Crossbars", *Proc. 2005 IEEE Infocom.*

[Cicconetti 2006] C. Cicconetti, L. Lenzini, A. Mingozi, K. Eklund, "Quality of Service Support in 802.16 Networks", *IEEE Network Magazine,* marzo/abril 2006, págs. 50–55.

[Cisco 12000 2009] Cisco Systems Inc., "Cisco XR 12000 Series and Cisco 12000 Series Routers", http://www.cisco.com/en/US/products/ps6342/index.html

[Cisco 8500 2009] Cisco Systems Inc., "Catalyst 8500 Campus Switch Router Architecture", http://www.cisco.com/univercd/cc/td/doc/product/l3sw/8540/rel_12_0/w5_6f/softcnfg/1cfg8500.pdf

[Cisco NAT 2009] Cisco Systems Inc., "How NAT Works", http://www.cisco.com/en/US/tech/tk648/tk361/technologies_tech_note09186a0080094831.shtml

[Cisco NAPA 2009] Cisco Systems Inc., "Cisco Network Application Performance Analysis (NAPA) Solution", http://www.cisco.com/en/US/products/sw/netmgtsw/index.html

[Cisco QoS 2009] Cisco Systems Inc., "Advanced QoS Services for the Intelligent Internet", http://www.cisco.com/warp/public/cc/pd/iosw/ioft/ioqo/tech/qos_wp.htm

[Cisco Queue 2009] Cisco Systems Inc., "Congestion Management Overview" http://www .cisco.com/en/US/docs/ios/12_2/qos/configuration/guide/qcfconmg.html

[Cisco Security 2009] Cisco Systems Inc., "Why You Need a Firewall", http://www.cisco .com/en/US/products/sw/secursw/ps743/products_user_guide_chapter09186a008007f303.html

[Cisco Switches 2009] Cisco Systems Inc, "Multiservice Switches", http://www.cisco.com/warp/public/cc/pd/si/index.shtml

[Cisco SYN 2009] Cisco Systems Inc., "Defining Strategies to Protect Against TCP SYN Denial of Service Attacks", http://www.cisco.com/en/US/tech/tk828/technologies_tech_note09186a00800f67d5.shtml

[Clark 1988] D. Clark, "The Design Philosophy of the DARPA Internet Protocols", *Proc. 1988 ACM SIGCOMM* (Stanford, CA, agosto 1988).

[Clarke 2002] I. Clarke, T. W. Hong, S. G. Miller, O. Sandberg, B. Wiley, "Protecting Free Expression Online with Freenet", *IEEE Internet Computing*, enero–febrero. 2002, págs. 40–49.

[Cohen 1977] D. Cohen, "Issues in Transnet Packetized Voice Communication", *Proc. Fifth Data Communications Symposium*, (Snowbird, Utah, septiembre 1977) págs. 6–13.

[Cohen 2003] B. Cohen, "Incentives to Build Robustness in BitTorrent", *First Workshop on the Economics of Peer-to-Peer Systems,* Berkeley, CA, junio 2003.

[Cookie Central 2009] Página principal de Cookie Central: http://www.cookiecentral.com/n_cookie_faq.htm

[CoolStreaming 2005] X. Zhang, J. Liu, J., B. Li y T.-S. P. Yum, "CoolStreamingDONet/: A Data-driven Overlay Network for Peer-to-Peer Live Media Streaming", *Proc. IEEE INFOCOM* (marzo 2005, Miami FL).

[Cormen 2001] T. H. Cormen, *Introduction to Algorithms,* 2nd Ed., MIT Press, Cambridge, MA, 2001.

[Crow 1997] B. Crow, I. Widjaja, J. Kim, P. Sakai, "IEEE 802.11 Wireless Local Area Networks", *IEEE Communications Magazine,* septiembre 1997, págs. 116–126.

[Crowcroft 1995] J. Crowcroft, Z. Wang, A. Smith, J. Adams, "A Comparison of the IETF and ATM Service Models", *IEEE Communications Magazine,* nov./ dic. 1995, págs. 12–16.

[Crowcroft 1999] J. Crowcroft, M. Handley, I. Wakeman, *Internetworking Multimedia,* Morgan-Kaufman, San Francisco, 1999.

[Cusumano 1998] M. A. Cusumano, D. B. Yoffie, *Competing on Internet Time: Lessons from Netscape and its Battle with Microsoft,* Free Press, NY, NY, 1998.

[Daigle 1991] J. N. Daigle, *Queuing Theory for Telecommunications,* Addison-Wesley, Reading, MA, 1991.

[Dalal 1978] Y. Dalal, R. Metcalfe, "Reverse Path Forwarding of Broadcast Packets", *Communications of the ACM*, Vol. 21, Nº 12 (diciembre 1978), págs. 1040–1048.

[Davie 2000] B. Davie y Y. Rekhter, *MPLS: Technology and Applications,* Morgan Kaufmann Series in Networking, 2000.

[Davies 2004] G. Davies, F. Kelly, "Network Dimensioning, Service Costing, and Pricing in a Packet-switched Environment", *Telecommunications Policy*, Vol. 28 (Nº 4), págs. 391–412.

[DEC 1990] Digital Equipment Corporation, "In Memoriam: J. C. R. Licklider 1915–1990", SRC Research Report 61, agosto 1990. http://www.memex.org/licklider.pdf

[DeClercq 2002] J. DeClercq, O. Paridaens, "Scalability Implications of Virtual Private Networks", *IEEE Communications Magazine*, Vol. 40 Nº 5 (mayo 2002), págs. 151–157.

[Demers 1990] A. Demers, S. Keshav, S. Shenker, "Analysis and Simulation of a Fair Queuing Algorithm", *Internetworking: Research and Experience,* Vol. 1, Nº 1, 1990, págs. 3–26.

[Denning 1997] D. Denning (Editor), P. Denning (Preface), *Internet Besieged: Countering Cyberspace Scofflaws,* Addison-Wesley, Reading, MA, 1997.

[dhc 2009] Página principal del grupo de trabajo Configuración dinámica de hosts de IETF: http://www.ietf.org/html.charters/dhc-charter.html

[Diggavi 2004] S. N. Diggavi, N. Al-Dhahir, A. Stamoulis, R. Calderbank, "Great Expectations: The Value of Spatial Diversity in Wireless Networks", *Proceedings of the IEEE,* vol. 92, Nº 2, febrero 2004.

[Diot 2000] C. Diot, B. N. Levine, B. Lyles, H. Kassem, D. Balensiefen, "Deployment Issues for the IP Multicast Service and Architecture", *IEEE Network,* Vol. 14, Nº 1 (enero/febrero 2000), págs. 78–88.

[Dmitiropoulos 2007] X. Dmitiropoulos, D. Krioukov, M. Fomenkov, B. Huffaker, Y. Hyun, KC Claffy, G. Riley, "AS Relationships: Inference and Validation", *ACM Computer Communication Review*, enero 2007.

[Dodge 2009] M. Dodge, "An Atlas of Cyberspaces", http://www.cybergeography.org/atlas/isp_maps.html

[Donahoo 2001] M. Donahoo, K. Calvert, *TCP/IP Sockets in C: Practical Guide for Programmers*, Morgan Kaufman, 2001.

[Doucer 2002] J. R. Douceur, "The Sybil Attack", *First International Workshop on Peer-to-Peer Systems (IPTPS '02)* (Cambridge, MA, marzo 2002).

[DSL 2009] Página principal de DSL Forum, http://www.dslforum.org/

[Dhunghel 2008] P. Dhungel, D. Wu, B. Schonhorst, K.W. Ross, "A Measurement Study of Attacks on BitTorrent Leechers", *7th International Workshop on Peer-to-Peer Systems (IPTPS 2008)* (Tampa Bay, febrero 2008).

[Droms 2002] R. Droms, T. Lemon, *The DHCP Handbook* (2nd Edition), SAMS Publishing, 2002.

[Edney 2003] J. Edney y W. A. Arbaugh, *Real 802.11 Security: Wi-Fi Protected Access and 802.11i,* Addison-Wesley Professional, 2003.

[Eklund 2002] K. Eklund, R. Marks, K. Stanswood, S. Wang, "IEEE Standard 802.16: A Technical Overview of the Wireless MAN Air Interface for Broadband Wireless Access", *IEEE Communications Magazine,* junio 2002, págs. 98–107.

[Ellis 1987] H. Ellis, "The Story of Non-Secret Encryption", http://jya.com/ellisdoc.htm

[Ericsson 2009] Ericsson, "The Evolution of Edge", http://www.ericsson.com/technology/whitepapers/broadband/evolution_of_EDGE.shtml

[ESM 2009] Página principal de End System Multicast, http://esm.cs.cmu.edu/

[Estrin 1997] D. Estrin, M. Handley, A. Helmy, P. Huang, D. Thaler, "A Dynamic Bootstrap Mechanism for Rendezvous-based Multicast Routing", P*roceedings of IEEE Infocom '98* (Nueva York, NY, abril 1998).

[Falkner 2007] J. Falkner, M. Piatek, J.P. John, A. Krishnamurthy, T. Anderson, "Profiling a Million Sser DHT", *Proc. ACM Internet Measurement Conference,* noviembre 2007.

[Faloutsos 1999] C. Faloutsos, M. Faloutsos, P. Faloutsos, "What Does the Internet Look Like? Empirical Laws of the Internet Topology", *Proc. 1999 ACM SIGCOMM* (Boston, MA, agosto 1999).

[Feamster 2004] N. Feamster, J. Winick, J. Rexford, "A Model for BGP Routing for Network Engineering", *Proc. 2004 ACM SIGMETRICS* (NY, NY, junio 2004).

[Feldman 2005] M. Feldman J. Chuang, "Overcoming Free-riding Behavior in Peer-to-peer Systems", *ACM SIGecom Exchanges*, julio 2005.

[Feldmeier 1988] D. Feldmeier, "Improving Gateway Performance with a Routing Table Cache", *Proc. 1988 IEEE Infocom* (Nueva Orleans LA, marzo 1988).

[Feldmeier 1995] D. Feldmeier, "Fast Software Implementation of Error Detection Codes", *IEEE/ACM Transactions on Networking,* Vol. 3, Nº 6 (diciembre 1995), págs. 640–652.

[FIPS 1995] Federal Information Processing Standard, "Secure Hash Standard", FIPS Publication 180-1. http://www.itl.nist.gov/fipspubs/fip180-1.htm

[Floyd 1999] S. Floyd, K. Fall, "Promoting the Use of End-to-End Congestion Control in the Internet", *IEEE/ACM Transactions on Networking,* Vol. 6, Nº 5 (octubre 1998), págs. 458–472.

[Floyd 2000] S. Floyd, M. Handley, J. Padhye, J. Widmer, "Equation-Based Congestion Control for Unicast Applications", *Proc. 2000 ACM SIGCOMM* (Estocolmo, Suecia, agosto 2000).

[Floyd 2001] S. Floyd, "A Report on Some Recent Developments in TCP Congestion Control", *IEEE Communications Magazine* (abril 2001),

[Floyd 2009] S. Floyd, "References on RED (Random Early Detection) Queue Management", http://www.icir.org/floyd/red.html

[Floyd Synchronization 1994] S. Floyd, V. Jacobson, "Synchronization of Periodic Routing Messages", *IEEE/ACM Transactions on Networking,* Vol. 2, Nº 2 (abril 1997), págs. 122–136.

[Floyd TCP 1994] S. Floyd, “TCP and Explicit Congestion Notification”, *ACM SIGCOMM Computer Communications Review,* Vol. 24, Nº 5, págs. 10–23, Oct. 1994.

[Fluhrer 2001] S. Fluhrer, I. Mantin, A. Shamir, “Weaknesses in the Key Scheduling Algorithm of RC4”, *Eighth Annual Workshop on Selected Areas in Cryptography* (Toronto, Canadá, agosto 2002).

[Fortz 2000] B. Fortz, M. Thorup, “Internet Traffic Engineering by Optimizing OSPF Weights”, *Proc. 2000 IEEE Infocom* (Tel Aviv, Israel, abril 2000).

[Fortz 2002] B. Fortz, J. Rexford, M. Thorup, “Traffic Engineering with Traditional IP Routing Protocols”, *IEEE Communication Magazine,* Oct. 2002.

[Foster 2002] I. Foster, “The Grid: A New Infrastructure for 21st Century Science”, *Physics Today*, 55(2):42–47, 2002.

[Fraleigh 2003] C. Fraleigh, T. Tobagi, C. Diot, “Provisioning IP backbone Networks to Support Latency Sensitive Traffic”, *Proc. IEEE Infocom Conference* (San Francisco, marzo 2003).

[France Telecom 2009] Object Identifier (OID) repository, http://asn1.elibel.tm.fr/oid/

[Fraleigh 2003] C. Fraleigh, F. Tobagi, C. Diot, “Provisioning IP Backbone Networks to Support Latency Sensitive Traffic”, *Proc. 2003 IEEE Infocom* (San Francisco, CA, marzo 2003).

[Freedman 2004] M. J. Freedman, E. Freudenthal, D. Mazires, “Democratizing Content Publication with Coral”, *USENIX NSDI,* 2004.

[Friedman 1999] T. Friedman, D. Towsley “Multicast Session Membership Size Estimation”, *Proc. 1999 IEEE Infocom* (Nueva York, USA, marzo 1999).

[Frost 1994] J. Frost, “BSD Sockets: A Quick and Dirty Primer”, http://world.std.com/~jimf/papers/sockets/sockets.html

[Gallagher 1983] R. G. Gallagher, P. A. Humblet, P. M. Spira, “A Distributed Algorithm for Minimum Weight-Spanning Trees”, *ACM Trans. on Programming Languages and Systems,* 1(5), (enero 1983), págs. 66–77.

[Gao 2001] L. Gao, J. Rexford, “Stable Internet Routing Without Global Coordination”, *IEEE/ACM Trans. Networking,* Vol. 9, Nº 6 (diciembre 2001), págs. 681–692.

[Garces-Erce 2003] L. Garces-Erce, K. W. Ross, E. Biersack, P. Felber, G. Urvoy-Keller, “TOPLUS: Topology Centric Lookup Service”, *Fifth Int. Workshop on Networked Group Communications (NGC 2003)*, (Munich, septiembre 2003) http://cis.poly.edu/~ross/papers/TOPLUS.pdf

[Gartner 2003] F. C. Gartner, “A Survey of Self-Stabilizing Spanning-Tree Construction Algorithms”, *Technical Report IC/2003/38*, Swiss Federal Institute of Technology (EPFL), School of Computer and Communication Sciences, 10 de junio de 2003. http://ic2.epfl.ch/publications/documents/IC_TECH_REPORT_200338.pdf.

[Gauthier 1999] L. Gauthier, C. Diot y J. Kurose, “End-to-end Transmission Control Mechanisms for Multiparty Interactive Applications on the Internet”, *Proc. 1999 IEEE Infocom* (Nueva York, NY, abril 1999).

[Giacopelli 1990] J. Giacopelli, M. Littlewood, W. D. Sincoskie “Sunshine: A high performance self-routing broadband packet switch architecture”, *1990 International Switching Symposium.* An extended version of this paper appeared in *IEEE J. Selected. Areas in Communications*, Vol. 9, Nº 8 (octubre 1991), págs. 1289–1298.

[Girard 1990] A. Girard, *Routing and Dimensioning in Circuit-Switched Networks,* Addison-Wesley, Reading, MA, 1990.

[Glitho 1995] R. Glitho, S. Hayes (eds.), special issue on Telecommunications Management Network, *IEEE Communications Magazine,* Vol. 33, Nº 3 (marzo 1995).

[Glitho 1998] R. Glitho, “Contrasting OSI Systems Management to SNMP and TMN”, *Journal of Network and Systems Management,* Vol. 6, Nº 2 (junio 1998), págs. 113–131.

[Gnutella 2009] “The Gnutella Protocol Specification, v0.4” http://www9.limewire.com/developer/gnutella_protocol_0.4.pdf

[Goodman 1997] David J. Goodman, *Wireless Personal Communications Systems*, Prentice- Hall, 1997.

[Goralski 1999] W. Goralski, *Frame Relay for High-Speed Networks,* John Wiley, Nueva York, 1999.

[Goralski 2001] W. Goralski, *Optical Networking and WDM,* Osborne/McGraw-Hill, Berkeley, CA, 2001.

[Griffin 2009] T. Griffin, "Interdomain Routing Links", http://www.cl.cam.ac.uk/~tgg22/interdomain/

[Guha 2006] S. Guha, N. Daswani, R. Jain, "An Experimental Study of the Skype Peer-to-Peer VoIP System", *Proc. Fifth Int. Workshop on P2P Systems* (Santa Barbara, CA, 2006).

[Guo 2005] L. Guo, S. Chen, Z. Xiao, E. Tan, X. Ding, X. Zhang, "Measurement, Analysis, and Modeling of BitTorrent-like Systems", *ACM Internet Measurement Conference (IMC),* 2005.

[Gupta 1998] P. Gupta, S. Lin, N. McKeown. "Routing lookups in hardware at memory access speeds", *Proc. 1998 IEEE Infocom* (San Francisco, CA, abril 1998), págs. 1241–1248.

[Gupta 2001] P. Gupta, N. McKeown, "Algorithms for Packet Classification", *IEEE Network Magazine,* Vol. 15, Nº 2 (Mar./abril 2001), págs. 24–32.

[Ha 2008] Ha, S., Rhee, I., L. Xu, "CUBIC: A New TCP-Friendly High-Speed TCP Variant", *ACM SIGOPS Operating System Review*, 2008.

[Hain 2005] T. Hain, "A Pragmatic Report on IPv4 Address Space Consumption", *Internet Protocol Journal*, Vol. 8, Nº 3.

[Halabi 2000] S. Halabi, *Internet Routing Architectures,* 2ª Ed., Cisco Press, 2000.

[Halperin 2008] D. Halperin, T. Heydt-Benjamin, B. Ransford, S. Clark, B. Defend, W. Morgan, K. Fu, T. Kohno, W. Maisel, "Pacemakers and implantable cardiac defibrillators: Software radio attacks and zero-power defenses", *Proc. 29th Annual IEEE Symposium on Security and Privacy*, mayo 2008.

[Hamada 1997] T. Hamada, H. Kamata, S. Hogg, "An Overview of the TINA Management Architecture", *Journal of Network and Systems Management,* Vol. 5. Nº 4 (diciembre 1997); págs. 411–435.

[Hei 2007] X. Hei, C. Liang, J. Liang, Y. Liu, K. W. Ross, "A Measurement Study of a Large-scale P2P IPTV System", *IEEE Trans. on Multimedia,* diciembre 2007.

[Heidemann 1997] J. Heidemann, K. Obraczka, J. Touch, "Modeling the Performance of HTTP over Several Transport Protocols", *IEEE/ACM Transactions on Networking,* Vol. 5, Nº 5 (octubre 1997), págs. 616–630.

[Held 2001] G. Held, *Data Over Wireless Networks: Bluetooth, WAP, and Wireless LANs*, McGraw-Hill, 2001.

[Hersent 2000] O. Hersent, D. Gurle, J-P. Petit, *IP Telephony: Packet-Based Multimedia Communication Systems*, Pearson Education Limited, Edinburgh, 2000.

[Holland 2001] G. Holland, N. Vaidya, V. Bahl, "A Rate-Adaptive MAC Protocol for Multi-Hop Wireless Networks", *Proc. 2001 ACM Int. Conference of Mobile Computing and Networking (Mobicom01)* (Roma, Italia, julio 2001).

[Hollot 2002] C.V. Hollot, V. Misra, D. Towsley, W. Gong, "Analysis and design of controllers for AQM routers supporting TCP flows", *IEEE Transactions on Automatic Control,* Vol. 47, Nº 6 (junio 2002), págs. 945–959.

[Huang 2002] C. Haung, V. Sharma, K. Owens, V. Makam, "Building Reliable MPLS Networks Using a Path Protection Mechanism", *IEEE Communications Magazine,* Vol. 40, Nº 3 (marzo 2002), págs. 156–162.

[Huang 2005] Y. Huang, R. Guerin, "Does Over-Provisioning Become More or Less Efficient as Networks Grow Larger?", *Proc. IEEE Int. Conf. Network Protocols (ICNP)* (Boston MA, noviembre 2005).

[Huang 2007] C. Huang, Jin Li, K.W. Ross, "Can Internet VoD be profitable?", *Proc ACM SIGCOMM* (Kyoto, agosto 2007).

[Huang 2008] C. Huang, J. Li, Angela Wang and K.W. Ross, "Understanding Hybrid CDN-P2P: Why Limelight Needs its Own Red Swoosh", NOSSDAV 2008, Braunschweig, Alemania, mayo 2008.

[Huitema 1998] C. Huitema, *IPv6: The New Internet Protocol,* 2nd Ed., Prentice Hall, Englewood Cliffs, NJ, 1998.

[Huston 1999a] G. Huston, "Interconnection, Peering, and Settlements—Part I", *The Internet Protocol Journal,* Vol. 2, Nº 1 (marzo 1999).

[Huston 2001] G. Huston, "Analyzing the Internet BGP Routing Table", *The Internet Protocol Journ*al, Vol. 4, Nº 1 (marzo 2001).

[Huston 2004] G. Huston, "NAT Anatomy: A Look Inside Network Address Translators", *The Internet Protocol Journal,* Vol. 7, Nº 3 (septiembre 2004).

[Huston 2008a] G. Huston, "Confronting Ipv4 Address Exhaustion", http://www.potaroo .net/ispcol/2008-10/v4depletion.html

[Huston 2008b] G. Huston, G. Michaelson, "IPv6 Deployment: Just where are we?" http:// www.potaroo.net/ispcol/2008-04/ipv6.html

[IAB 2009] Página principal de Internet Architecture Board, http://www.iab.org/

[IANA 2009a] Página principal de Internet Assigned Number Authority, http://www.iana.org/

[IANA 2009b] Internet Assigned Number Authority, "Private Enterprise Numbers" http:// www.iana.org/assignments/enterprise-numbers

[IANA Protocol Numbers 2009] Internet Assigned Numbers Authority, Protocol Numbers, http://www.iana.org/assignments/protocol-numbers/protocol-numbers.xhtml

[ICANN 2009] Página principal de The Internet Corporation for Assigned Names and Numbers, http://www.icann.org

[IEC Optical 2009] IEC Online Education, "Optical Access", http://www.iec.org/online/tutorials/opt_acc/

[IEEE 802 2009] Página principal del comité de estándares IEEE 802 LAN/MAN: http://www.ieee802.org/

[IEEE 802.11 1999] IEEE 802.11, 1999 Edition (ISO/IEC 8802-11: 1999) IEEE Standards for Informa-tion Technology—Telecommunications and Information Exchange Between Systems—Local and Metro-politan Area Network—Specific Requirements—Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specification, http://standards.ieee.org/getieee802/download/802.11-1999.pdf

[IEEE 802.11n] IEEE, "IEEE P802.11—Task Group N—Meeting Update: Status of 802.11n" http://grouper.ieee.org/groups/802/11/Reports/tgn_update.htm

[IEEE 802.15 2009] Página principal del grupo de trabajo IEEE 802.15 para WPAN: http://grouper.ieee.org/groups/802/15/.

[IEEE 802.16d 2004] IEEE, "IEEE Standard for Local and metropolitan area networks, Part 16: Air Interface for Fixed Broadband Wireless Access Systems", http://standards.ieee.org/getieee802/download/802.16-2004.pdf

[IEEE 802.16e 2005] IEEE, "IEEE Standard for Local and metropolitan area networks, Part 16: Air Interface for Fixed and Mobile Broadband Wireless Access Systems, Amendment 2: Physical and Medium Access Control Layers for Combined Fixed and Mobile Operation in Licensed Bands and Corrigendum 1", http://standards.ieee.org/getieee802/download/802 .16e-2005.pdf

[IEEE 802.1q 2005] IEEE, "IEEE Standard for Local and Metropolitan Area Networks: Virtual Bridged Local Area Networks", http://standards.ieee.org/getieee802/download/802 .1Q-2005.pdf

[IEEE 802.1X] IEEE Std 802.1X-2001 Port-Based Network Access Control, http:// standards.ieee.org/reading/ieee/std_public/description/lanman/802.1x-2001_desc.html

[IEEE 802.3 2009] IEEE, "IEEE 802.3 CSMA/CD (Ethernet)," http://grouper.ieee.org/groups/802/3/

[IEEE 802.5 2009] Página principal de IEEE, IEEE 802.5, http://www.ieee802.org/5/www8025org/

[IETF 2009] Página principal de Internet Engineering Task Force, http://www.ietf.org

[IMAP 2009] The IMAP Connection, http://www.imap.org/

[Intel 2009] Intel Corp, “Intel® 82544 Gigabit Ethernet Controller” http://www.intel.com/design/network/products/lan/docs/82544_docs.htm

[Intel WiMax 2009] Intel Corp., “WiMax Technology”, http://www.intel.com/technology/wimax/index.htm

[Internet2 Multicast 2009] Página principal de Internet2 Multicast Working Group, http://multicast.internet2.edu/

[IPv6 2009] Página principal de IPv6.com, http://www.ipv6.com/

[ISC 2009] Página principal de Internet Systems Consortium, http://www.isc.org

[ISI 1979] Information Sciences Institute, “DoD Standard Internet Protocol”, Internet Engineering Note 123, diciembre 1979. http://www.isi.edu/in-notes/ien/ien123.txt

[ISO 2009] Página principal de la Organización Internacional de Estandarización (ISO, International Organization for Standardization): http://www.iso.org/

[ISO X.680 2002] International Organization for Standardization, “X.680: ITU-T Recommendation X.680 (2002) Information Technology—Abstract Syntax Notation One (ASN.1): Specification of Basic Notation.” http://www.itu.int/ITU-T/studygroups/com17/languages/X.680-0207.pdf

[ITU 2005] International Telecommunication Union, *The Internet of Things,* 2005, http:// www.itu.int/osg/spu/publications/internetofthings/InternetofThings_summary.pdf

[ITU 2009] Página principal de ITU, http://www.itu.int/

[ITU Statistics 2009] International Telecommunications Union, “ICT Statistics”, http:// www.itu.int/ITU-D/icteye/Reports.aspx

[Iyer 2002] S. Iyer, R. Zhang, N. McKeown, “Routers with a Single Stage of Buffering”, *Proc. 2002 ACM SIGCOMM* (Pittsburgh, PA, agosto 2002).

[Jacobson 1988] V. Jacobson, “Congestion Avoidance and Control”, *Proc. 1988 ACM SIGCOMM* (Stanford, CA, agosto 1988), págs. 314–329.

[Jain 1986] R. Jain, “A timeout-based congestion control scheme for window flow-controlled networks”, *IEEE Journal on Selected Areas in Communications SAC-4*, 7 (octubre 1986).

[Jain 1989] R. Jain, “A Delay-Based Approach for Congestion Avoidance in Interconnected Heterogeneous Computer Networks”, *ACM SIGCOMM Computer Communications Review,* Vol. 19, Nº 5 (1989), págs. 56–71.

[Jain 1994] R. Jain, *FDDI Handbook: High-Speed Networking Using Fiber and Other Media,* Addison-Wesley, Reading, MA, 1994.

[Jain 1996] R. Jain. S. Kalyanaraman, S. Fahmy, R. Goyal, S. Kim, “Tutorial Paper on ABR Source Behavior”, *ATM Forum*/96-1270, Oct. 1996. http://www.cse.wustl.edu/~jain/atmf/ftp/atm96-1270.pdf

[Jaiswal 2003] S. Jaiswal, G. Iannaccone, C. Diot, J. Kurose, D. Towsley, “Measurement and Classification of Out-of-Sequence Packets in a Tier-1 IP backbone”, *Proc. 2003 IEEE Infocom.*

[Jakobson 1993] G. Jacobson, M. Weissman, “Alarm Correlation”, *IEEE Network Magazine,* 1993, págs. 52–59.

[Ji 2003] P. Ji, Z. Ge, J. Kurose, D. Towsley, “A Comparison of Hard-state and Soft-state Signaling Protocols”, *Proc. 2003 ACM SIGCOMM* (Karlsruhe, Alemania, agosto 2003).

[Jiang 2001] W. Jiang, J. Lennox, H. Schulzrinne, K. Singh, “Towards Junking the PBX: Deploying IP Telephony”, *NOSSDAV’01* (Port Jefferson, NY, junio 2001).

[Jin 2004] C. Jin, D. X. We, S. Low, “FAST TCP: Motivation, architecture, algorithms, performance”, *Proc. 2004 IEEE Infocom* (Hong Kong, marzo 2004).

[Juniper 2009] Juniper Networks, “Provider Network Management”, http://www.juniper .net/solutions/service_provider/provider_network_management/

[Kaaranen 2001] H. Kaaranen, S. Naghian, L. Laitinen, A. Ahtiainen, V. Niemi, *Networks: Architecture, Mobility and Services,* Nueva York: John Wiley & Sons, 2001.

[Kahn 1978] R. E. Kahn, S. Gronemeyer, J. Burchfiel, R. Kunzelman, "Advances in Packet Radio Technology", *Proc. of the IEEE*, 66, 11 (noviembre 1978).

[Kamerman 1997] A. Kamerman, L. Monteban, "WaveLAN-II: A High–Performance Wireless LAN for the Unlicensed Band", *Bell Labs Technical Journal,* Summer 1997, págs. 118–133.

[Kangasharju 2000] J. Kangasharju, K. W. Ross, J. W. Roberts, "Performance Evaluation of Redirection Schemes in Content Distribution Networks," *Proc. 5th Web Caching and Content Distribution Workshop* (Lisboa, Portugal, mayo 2000).

[Kar 2000] K. Kar, M. Kodialam, T. V. Lakshman, "Minimum Interference Routing of Bandwidth Guaranteed Tunnels with MPLS Traffic Engineering Applications", *IEEE J. Selected Areas in Communications*, diciembre 2000.

[Karol 1987] M. Karol, M. Hluchyj, A. Morgan, "Input Versus Output Queuing on a Space- Division Packet Switch", *IEEE Transactions on Communications,* Vol. 35, Nº 12, (diciembre 1987), págs. 1347–1356.

[Katabi 2002] D. Katabi, M. Handley, C. Rohrs, "Internet Congestion Control for Future High Bandwidth-Delay Product Environments", *Proc. 2002 ACM SIGCOMM* (Pittsburgh, PA, agosto 2002).

[Katzela 1995] I. Katzela, M. Schwartz. "Schemes for Fault Identification in Communication Networks", *IEEE/ACM Transactions on Networking,* Vol. 3, Nº 6 (diciembre 1995), págs. 753–764.

[Kaufman 1995] C. Kaufman, R. Perlman, M. Speciner, *Network Security, Private Communication in a Public World,* Prentice Hall, Englewood Cliffs, NJ, 1995.

[Kelly 1998] F. P. Kelly, A. Maulloo, D. Tan, "Rate control for communication networks: Shadow prices, proportional fairness and stability", *J. Operations Res. Soc.,* Vol. 49, Nº 3, págs. 237–252, marzo 1998.

[Kelly 2003] T. Kelly, "Scalable TCP: improving performance in high speed wide area networks", *ACM SIGCOMM Computer Communications Review,* Vol. 33, Nº 2 (abril 2003), págs. 83–91.

[Keshav 1998] S. Keshav, R. Sharma, "Issues and Trends in Router Design", *IEEE Communications Magazine,* Vol. 36, Nº 5 (mayo 1998), págs. 144–151.

[Keslassy 2003] I. Keslassy, S. Chuang, K. Yu, D. Miller, M. Horowitz, O. Solgaard, McKeown, "Scaling Internet Routers Using Optics", *Proc. 2003 ACM SIGCOMM* (Karlsruhe, Alemania, agosto 2003).

[Kilkki 1999] K. Kilkki, *Differentiated Services for the Internet*, Macmillan Technical Publishing, Indianapolis, IN, 1999.

[Kim 2005] H. Kim, S. Rixner, V. Pai, "Network Interface Data Caching", *IEEE Transactions on Computers*, Vol. 54, Nº 11 (noviembre 2005), págs. 1394–1408.

[Kim 2008] C. Kim, M. Caesar, J. Rexford, "Floodless in SEATTLE: A Scalable Ethernet Architecture for Large Enterprises", *Proc.* ACM SI*GCOMM '08* (agosto 2008, Seattle, WA).

[Kleinrock 1961] L. Kleinrock, "Information Flow in Large Communication Networks", RLE Quarterly Progress Report, julio 1961.

[Kleinrock 1964] L. Kleinrock, *1964 Communication Nets: Stochastic Message Flow and Delay,* McGraw-Hill, NY, NY, 1964.

[Kleinrock 1975] L. Kleinrock, *Queuing Systems, Vol. 1,* John Wiley, Nueva York, 1975.

[Kleinrock 1975b] L. Kleinrock, F. A. Tobagi, "Packet Switching in Radio Channels: Part I—Carrier Sense Multiple-Access Modes and Their Throughput-Delay Characteristics", *IEEE Transactions on Communications,* Vol. 23, Nº 12 (diciembre 1975), págs. 1400–1416.

[Kleinrock 1976] L. Kleinrock, *Queuing Systems, Vol. 2,* John Wiley, Nueva York, 1976.

[Kleinrock 2004] L. Kleinrock, "The Birth of the Internet", http://www.lk.cs.ucla.edu/LK/Inet/birth.html

[Kohler 2006] E. Kohler, M. Handley, S. Floyd, "DDCP: Designing DCCP: Congestion Control Without Reliability", *Proc. 2006 ACM SIGCOMM* (Pisa, Italia, septiembre 2006).

[Korhonen 2003] J. Korhonen, *Introduction to 3G Mobile Communications*, 2ª ed., Artech House, 2003.

[Koziol 2003] J. Koziol, *Intrusion Detection with Snort,* Sams Publishing, 2003.

[Krishnamurthy 2001] B. Krishnamurthy y J. Rexford, *Web Protocols and Practice: HTTP/ 1.1, Networking Protocols, and Traffic Measurement*, Addison-Wesley, Boston, MA, 2001.

[Krishnamurthy 2001b] B. Krishnamurthy, C. Wills, Y. Zhang, "On the Use and Performance of Content Distribution Networks", *ACM Internet Measurement Conference,* 2001.

[Kulkarni 2005] S. Kulkarni, C. Rosenberg, "Opportunistic Scheduling: Generalizations to Include Mul-tiple Constraints, Multiple Interfaces, and Short Term Fairness", *Wireless Networks*, 11, 557–569, 2005.

[Kumar 2006] R. Kumar, K.W. Ross, "Optimal Peer-Assisted File Distribution: Single and Multi-Class Problems", *IEEE Workshop on Hot Topics in Web Systems and Technologies*, Boston, 2006.

[Kurose 1996] Unix Network Progranmming, http://gaia.cs.umass.edu/ntu_socket/

[Labovitz 1997] C. Labovitz, G. R. Malan, F. Jahanian, "Internet Routing Instability", *Proc. 1997 ACM SIGCOMM* (Cannes, Francia, septiembre 1997), págs. 115–126.

[Labrador 1999] M. Labrador, S. Banerjee, "Packet Dropping Policies for ATM and IP Networks", *IEEE Communications Surveys*, Vol. 2, Nº 3 (tercer trimestre de 1999), págs. 2–14.

[Lacage 2004] M. Lacage, M.H. Manshaei, T. Turletti, "IEEE 802.11 Rate Adaptation: A Practical Approach", *ACM Int. Symposium on Modeling, Analysis, and Simulation of Wireless and Mobile Systems (MSWiM)* (octubre 2004, Venecia, Italia).

[Lakshman 1997] T. V. Lakshman, U. Madhow, "The Performance of TCP/IP for Networks with High Bandwidth-Delay Products and Random Loss", *IEEE/ACM Transactions on Networking,* Vol. 5 Nº 3 (1997). págs. 336–350.

[Lam 1980] S. Lam, "A Carrier Sense Multiple Access Protocol for Local Networks", *Computer Networks,* Vol. 4 (1980), págs. 21–32, 1980.

[Lamport 1981] L. Lamport, "Password Authentication with Insecure Communication", *Communications of the ACM,* Vol. 24, Nº 11 (noviembre 1981), págs. 770–772.

[Larmouth 1996] J. Larmouth, *Understanding OSI,* International Thomson Computer Press 1996. Chapter 8 of this book deals with ASN.1 and is available on-line at http://www.salford .ac.uk/iti/books/osi/all.html#head8

[Lawton 2001] G. Lawton, "Is IPv6 Finally Gaining Ground?" *IEEE Computer Magazine* (agosto 2001), págs. 11–15.

[Leiner 1998] B. Leiner, V. Cerf, D. Clark, R. Kahn, L. Kleinrock, D. Lynch, J. Postel, L. Roberts, S. Woolf, "A Brief History of the Internet", http://www.isoc.org/internet/history/brief.html

[Li 2004] L. Li, D. Alderson, W. Willinger, J. Doyle, "A First-Principles Approach to Understanding the Internet's Router-Level Topology", *Proc. 2004 ACM SIGCOMM* (Portland, Oregón, agosto 2004).

[Li 2007] J. Li, M. Guidero, Z. Wu, E. Purpus, T. Ehrenkranz, "BGP Routing Dynamics Revisited." *ACM Computer Communication Review,* abril 2007.

[Liang 2006] J. Liang, N. Naoumov, K.W. Ross, "The Index Poisoning Attack in P2P File-Sharing Systems", *Proc. 2006 IEEE Infocom 2006* (Barcelona, España, abril 2006).

[Lin 2001] Y. Lin, I. Chlamtac, *Wireless and Mobile Network Architectures*, John Wiley and Sons, Nueva York, NY, 2001.

[Liogkas 2006] N. Liogkas, R. Nelson, E. Kohler, L. Zhang, "Exploiting BitTorrent For Fun (But Not Profit)," *6th International Workshop on Peer-to-Peer Systems (IPTPS 2006).*

[Liu 2002] B. Liu, D. Goeckel, D. Towsley, "TCP-Cognizant Adaptive Forward Error Correction in Wireless Networks", *Proc. 2002 Global Internet.*

[Locher 2006] T. Locher, P. Moor, S. Schmid, R. Wattenhofer, "Free Riding in BitTorrent is Cheap", *Proc. ACM HotNets 2006* (Irvine CA, noviembre 2006).

[Lui 2004] J. Lui, V. Misra, D. Rubenstein, "On the Robustness of Soft State Protocols", *Proc. IEEE Int. Conference on Network Protocols (ICNP '04*), págs. 50–60.

[Luotonen 1998] A. Luotonen, *Web Proxy Servers,* Prentice Hall, Englewood Cliffs, Nueva Jersey, 1998.

[Lynch 1993] D. Lynch, M. Rose, *Internet System Handbook,* Addison-Wesley, Reading, MA, 1993.

[Macedonia 1994] M. Macedonia, D. Brutzman, "MBone Provides Audio and Video Across the Internet", *IEEE Computer Magazine,* Vol. 27, Nº 4 (abril 1994), págs. 30–36.

[Mahdavi 1997] J. Mahdavi, S. Floyd, "TCP-Friendly Unicast Rate-Based Flow Control", unpublished note, enero 1997.

[Malware 2006] Computer Economics, "2005 Malware Report: The Impact of Malicious Code Attacks", http://www.computereconomics.com

[manet 2009] IETF Mobile Ad-hoc Networks (manet) Working Group, http://www.ietf.org/html.charters/manet-charter.html

[Mao 2002] Z. M. Mao, C. Cranor, F. Douglis, M. Rabinovich, O. Spatscheck, J. Wang, "A Precise and Efficient Evaluation of the Proximity between Web Clients and Their Local DNS Servers", *USENIX 2002.*

[Maymounkov 2002] P. Maymounkov, D. Mazières. "Kademlia: A Peer-to-Peer Information System Based on the XOR Metric." *Proceedings of the 1st International Workshop on Peerto-Peer Systems (IPTPS '02)*, págs. 53–65, marzo 2002.

[McKeown 1997a] N. McKeown, M. Izzard, A. Mekkittikul, W. Ellersick, M. Horowitz, "The Tiny Tera: A Packet Switch Core", *IEEE Micro Magazine,* enero/febrero 1997.

[McKeown 1997b] N. McKeown, "A Fast Switched Backplane for a Gigabit Switched Router", *Business Communications Review,* Vol. 27, Nº 12. http://tiny-tera.stanford.edu/~nickm/papers/cisco_fasts_wp.pdf

[McQuillan 1980] J. McQuillan, I. Richer, E. Rosen, "The New Routing Algorithm for the Arpanet", *IEEE Transactions on Communications,* Vol. 28, Nº 5 (mayo 1980), págs. 711–719.

[Medhi 1997] D. Medhi, D. Tipper (eds.), Special Issue: Fault Management in Communication Networks, *Journal of Network and Systems Management,* Vol. 5. Nº 2 (junio 1997).

[Meng 2005] X. Meng, "IPv4 Address Allocation and the BGP Routing Table Evolution", *Computer Communication Reviews,* Vol. 35, Nº 1 (2005), págs. 71–80.

[Metcalfe 1976] R. M. Metcalfe, D. R. Boggs. "Ethernet: Distributed Packet Switching for Local Computer Networks", *Communications of the Association for Computing Machinery,* Vol. 19, Nº 7, (julio 1976), págs. 395–404.

[MFA Forum 2009] Página principal de Ip/MPLS Forum: http://www.ipmplsforum.org/

[Microsoft Player Media 2009] Página principal de Microsoft Windows Media, http://www.microsoft.com/windows/windowsmedia/

[Miller 1997] M.A. Miller, *Managing Internetworks with SNMP,* 2ª ed., M & T Books, Nueva York, 1997.

[Mirkovic 2005] J. Mirkovic, S. Dietrich, D. Dittrich. P. Reiher, *Internet Denial of Service: Attack and Defense Mechanisms*, Prentice Hall, 2005.

[Mockapetris 1988] P. V. Mockapetris, K. J. Dunlap, "Development of the Domain Name System", *Proc. 1988 ACM SIGCOMM* (Stanford, CA, agosto 1988).

[Mockapetris 2005] P. Mockapetris, Sigcomm Award Lecture, vídeo disponible en http:// www.postel.org/sigcomm

[Mogul 2003] J. Mogul, "TCP offload is a dumb idea whose time has come". *Proc. HotOS IX: The 9th Workshop on Hot Topics in Operating Systems*, (2003) USENIX Association.

[Molinero-Fernández 2002] P. Molinero-Fernández, N. McKeown, H. Zhang, "Is IP Going to Take Over the World (of Communications)?", *Proc. 2002 ACM Hotnets*.

[Molle 1987] M. L. Molle, K. Sohraby, A. N. Venetsanopoulos, "Space-Time Models of Asynchronous CSMA Protocols for Local Area Networks", *IEEE Journal on Selected Areas in Communications,* Vol. 5, Nº 6, (1987) págs. 956–968.

[Moore 2001] D. Moore, G. Voelker, S. Savage, "Inferring Internet Denial of Service Activity", *Proc. 2001 USENIX Security Symposium* (Washington DC, agosto 2001).

[Moore 2003] D. Moore, V. Paxson, S. Savage, C. Shannon, S. Staniford, N. Weaver, "Inside the Slammer Worm", *2003 IEEE Security and Privacy Conference.*

[Moshchuck 2006] A. Moshchuk, T. Bragin, S. Gribble, H. Levy, "A Crawler-based Study of Spyware on the Web", *Proc. 13th Annual Network and Distributed Systems Security Symposium (NDSS 2006)* (San Diego, CA, febrero 2006).

[Mouly 1992] M. Mouly, M. Pautet, *The GSM System for Mobile Communications*, Cell and Sys, Palaiseau, Francia, 1992.

[Moy 1998] J. Moy, *OSPF: Anatomy of An Internet Routing Protocol*, Addison-Wesley, Reading, MA, 1998.

[Mukherjee 1997] B. Mukherjee, *Optical Communication Networks*, McGraw-Hill, 1997.

[Murphy 2003] Véase [RFC 4272]

[Nahum 2002] E. Nahum, T. Barzilai, D. Kandlur, "Performance Issues in WWW Servers", *IEEE/ACM Transactions on Networking*, Vol 10, Nº 1 (Feb. 2002).

[Naoumov 2006] N. Naoumov, K.W. Ross, "Exploiting P2P Systems for DDoS Attacks", *Intl Workshop on Peer-to-Peer Information Management*, (Hong Kong, mayo 2006),

[Neglia 2007] G. Neglia, G. Reina, H. Zhang, D. Towsley, A. Venkataramani, J. Danaher, "Availability in BitTorrent Systems", *Proc. IEEE INFOCOM 2007,* mayo 2007.

[Neumann 1997] R. Neumann, "Internet Routing Black Hole", *The Risks Digest: Forum on Risks to the Public in Computers and Related Systems,* Vol. 19, Nº 12 (mayo 1997). http:// catless.ncl.ac.uk/Risks/19.12.html#subj1.1

[Newman 2008] D. Newman, "802.11n Gear 10 Times Faster Than Current Wi-Fi Offerings", *Network World,* Oct. 27, 2008. http://www.networkworld.com/reviews/2008/102708-wlan-test.html?page=1

[Nicholson 2006] A Nicholson, Y. Chawathe, M. Chen, B. Noble, D. Wetherall, "Improved Access Point Selection", *Proc. 2006 ACM Mobisys Conference* (Uppsala Sweden, 2006).

[Nielsen 1997] H. F. Nielsen, J. Gettys, A. Baird-Smith, E. Prud'hommeaux, H. W. Lie, C. Lilley, "Network Performance Effects of HTTP/1.1, CSS1, and PNG", *W3C Document,* 1997 (also appears in *Proc. 1997 ACM SIGCOM,* (Cannes, Francia, septiembre 1997), págs. 155–166.

[NIST 2001] National Institute of Standards and Technology, "Advanced Encryption Standard (AES)," Federal Information Processing Standards 197, noviembre 2001, http://csrc.nist .gov/publications/fips/fips197/fips-197.pdf

[Nmap 2009] Página principal de Nmap: http://www.insecure.com/nmap

[Nonnenmacher 1998] J. Nonnenmacher, E. Biersak, D. Towsley, "Parity-Based Loss Recovery for Reliable Multicast Transmission", *IEEE/ACM Transactions on Networking,* Vol. 6, Nº 4 (agosto 1998), págs. 349–361.

[NTIA 1998] National Telecommunications and Information Administration (NTIA), US Department of Commerce, "Management of Internet names and addresses", Docket Number: 980212036-8146-02. http://www.ntia.doc.gov/ntiahome/domainname/6_5_98dns.htm

[Odlyzko 2003] A. Odlyzko, "Internet Traffic Growth: Sources and Implications", A. M. Optical Transmission Systems and Equipment for WDM Networking II, *Proc. SPIE*, 5247, 2003, págs. 1–15. http://www.dtc.umn.edu/~odlyzko/doc/itcom.internet.growth.pdf.

[OSI 2009] Página principal de International Organization for Standardization, http://www.iso.org/iso/en/ISOOnline.frontpage

[OSS 2009] OSS Nokalva, “ASN.1 Resources”, http://www.oss.com/asn1/

[Padhye 2000] J. Padhye, V. Firoiu, D. Towsley, J. Kurose, “Modeling TCP Reno Performance: A Simple Model and its Empirical Validation”, *IEEE/ACM Transactions on Networking,* Vol. 8 Nº 2 (abril 2000), págs. 133–145.

[Padhye 2001] J. Padhye, S. Floyd, “On Inferring TCP Behavior”, *Proc. 2001 ACM SIGCOMM,* (San Diego, CA, agosto 2001).

[Pan 1997] P. Pan, H. Schulzrinne, “Staged Refresh Timers for RSVP”, *Proc. 2nd Global Internet Conference* (Phoenix, AZ, diciembre 1997).

[Parekh 1993] A. Parekh, R. Gallagher, “A generalized processor sharing approach to flow control in integrated services networks: the single-node case”, *IEEE/ACM Transactions on Networking,* Vol. 1, Nº 3 (junio 1993), págs. 344–357.

[Partridge 1992] C. Partridge, S. Pink, “An Implementation of the Revised Internet Stream Protocol (ST-2),” *Journal of Internetworking: Research and Experience*, Vol. 3, Nº. 1 (marzo 1992).

[Partridge 1998] C. Partridge, et al. “A Fifty Gigabit per second IP Router”, *IEEE/ACM Transactions on Networking,* Vol. 6, Nº 3 (junio 1998), págs. 237–248.

[Paxson 1997] V. Paxson, “End-to-end Internet packet dynamics”, *Proc. 1997 ACM SIGCOMM* (Cannes, Francia, septiembre 1997).

[Perkins 1994] A. Perkins, “Networking with Bob Metcalfe”, *The Red Herring Magazine,* noviembre 1994.

[Perkins 1998] C. Perkins, O. Hodson, V. Hardman, “A Survey of Packet Loss Recovery Techniques for Streaming Audio”, *IEEE Network Magazine,* sept./oct. 1998, págs. 40–47.

[Perkins 1998b] C. Perkins, *Mobile IP: Design Principles and Practice*, Addison-Wesley, Reading, MA, 1998.

[Perkins 2000] C. Perkins, *Ad Hoc Networking*, Addison-Wesley, Reading, MA, 2000.

[Perlman 1999] R. Perlman, *Interconnections: Bridges, Routers, Switches, and Internetworking Protocols,* 2ª ed., Addison-Wesley Professional Computing Series, Reading, MA, 1999.

[PGPI 2009] The International PGP Home Page, http://www.pgpi.org

[Phifer 2000] L. Phifer, “The Trouble with NAT”, *The Internet Protocol Journal,* Vol. 3, Nº 4 (diciembre 2000), http://www.cisco.com/warp/public/759/ipj_3-4/ipj_3-4_nat.html

[Piatek 2007] M. Piatek, T. Isdal, T. Anderson, A. Krishnamurthy, A. Venkataramani, “Do Incentives Build Robustness in Bittorrent?”, *Proc. NSDI,* 2007.

[Piatek 2008] M. Piatek, T. Isdal, A. Krishnamurthy, T. Anderson, “One hop Reputations for Peer-to-peer File Sharing Workloads”, *Proc. NSDI,* 2008.

[Pickholtz 1982] R. Pickholtz, D. Schilling, L. Milstein, “Theory of Spread Spectrum Communication—a Tutorial”, *IEEE Transactions on Communications,* Vol. 30, Nº 5 (mayo 1982), págs. 855–884.

[pingplotter 2009] Página principal de pingplotter, http://www.pingplotter.com

[Piscatello 1993] D. Piscatello, A. Lyman Chapin, *Open Systems Networking,* Addison-Wesley, Reading, MA, 1993.

[Point Topic 2006] Point Topic Ltd., *World Broadband Statistics Q1 2006,* http://www .pointtopic.com

[Potaroo 2009] “Growth of the BGP Table–1994 to Present”, http://bgp.potaroo.net/

[PPLive 2009] Página principal de PPLive: http://www.pplive.com

[PriMetrica 2009] Global Internet Geography, http://www.telegeography.com/products/gig/index.php

[QuickTime 2009] Página principal de QuickTime: http://www.apple.com/quicktime

[Quittner 1998] J. Quittner, M. Slatalla, *Speeding the Net: The Inside Story of Netscape and How it Challenged Microsoft,* Atlantic Monthly Press, 1998.

[Ramakrishnan 1990] K. K. Ramakrishnan, R. Jain, "A Binary Feedback Scheme for Congestion Avoidance in Computer Networks", *ACM Transactions on Computer Systems,* Vol. 8, Nº 2 (mayo 1990), págs. 158–181.

[Raman 1999] S. Raman, S. McCanne, "A Model, Analysis, and Protocol Framework for Soft State-based Communication", *Proc. 1999 ACM SIGCOMM* (Boston, MA, agosto 1999).

[Raman 2007] B. Raman, K. Chebrolu, "Experiences in using WiFi for Rural Internet in India", *IEEE Communications Magazine,* Special Issue on New Directions in Networking Technologies in Emerging Economies, Jan 2007.

[Ramaswami 1998] R. Ramaswami, K. Sivarajan, *Optical Networks: A Practical Perspective,* Morgan Kaufman Publishers, 1998.

[Ramjee 1994] R. Ramjee, J. Kurose, D. Towsley, H. Schulzrinne, "Adaptive Playout Mechanisms for Packetized Audio Applications in Wide-Area Networks", *Proc. 1994 IEEE Infocom.*

[Rao 1996] K. R. Rao and J. J. Hwang, *Techniques and Standards for Image, Video and Audio Coding,* Prentice Hall, Englewood Cliffs, NJ, 1996.

[RAT 2009] Robust Audio Tool, http://www-mice.cs.ucl.ac.uk/multimedia/software/rat/

[Ratnasamy 2001] S. Ratnasamy, P. Francis, M. Handley, R. Karp, S. Shenker, "A Scalable Content-Addressable Network", *Proc. 2001 ACM SIGCOMM* (San Diego, CA, agosto 2001).

[RealNetworks 2009] Página principal de RealNetworks: http://www.realnetworks.com

[Ren 2006] S. Ren, L. Guo y X. Zhang, "ASAP: an AS-aware peer-relay protocol for high quality VoIP", *Proc. 2006 IEEE ICDCS* (Lisboa, Portugal, julio 2006).

[Rescorla 2001] E. Rescorla, *SSL and TLS: Designing and Building Secure Systems,* Addison-Wesley, Boston, 2001.

[RFC 001] S. Crocker, "Host Software", RFC 001 (the *very first* RFC!).

[RFC 768] J. Postel, "User Datagram Protocol", RFC 768, agosto 1980.

[RFC 789] E. Rosen, "Vulnerabilities of Network Control Protocols", RFC 789.

[RFC 791] J. Postel, "Internet Protocol: DARPA Internet Program Protocol Specification", RFC 791, septiembre 1981.

[RFC 792] J. Postel, "Internet Control Message Protocol", RFC 792, septiembre 1981.

[RFC 793] J. Postel, "Transmission Control Protocol", RFC 793, septiembre 1981.

[RFC 801] J. Postel, "NCP/TCP Transition Plan", RFC 801, noviembre 1981.

[RFC 826] D. C. Plummer, "An Ethernet Address Resolution Protocol—or—Converting Network Protocol Addresses to 48 bit Ethernet Address for Transmission on Ethernet Hardware", RFC 826, noviembre 1982.

[RFC 829] V. Cerf, "Packet Satellite Technology Reference Sources", RFC 829, noviembre 1982.

[RFC 854] J. Postel, J. Reynolds, "TELNET Protocol Specification", RFC 854, mayo 1993.

[RFC 950] J. Mogul, J. Postel, "Internet Standard Subnetting Procedure", RFC 950, agosto 1985.

[RFC 959] J. Postel y J. Reynolds, "File Transfer Protocol (FTP)," RFC 959, octubre 1985.

[RFC 977] B. Kantor, P. Lapsley, "Network News Transfer Protocol", RFC 977, febrero 1986.

[RFC 1028] J. Davin, J.D. Case, M. Fedor, M. Schoffstall, "A Simple Gateway Monitoring Protocol", RFC 1028, noviembre 1987.

[RFC 1034] P. V. Mockapetris, "Domain Names—Concepts and Facilities", RFC 1034, noviembre 1987.

[RFC 1035] P. Mockapetris, "Domain Names—Implementation and Specification", RFC 1035, noviembre 1987.

[RFC 1058] C. L. Hendrick, “Routing Information Protocol”, RFC 1058, junio 1988.

[RFC 1071] R. Braden, D. Borman y C. Partridge, “Computing The Internet Checksum”, RFC 1071, septiembre 1988.

[RFC 1075] D. Waitzman, C. Partridge, S. Deering, “Distance Vector Multicast Routing Protocol”, RFC 1075, noviembre 1988.

[RFC 1112] S. Deering, “Host Extension for IP Multicasting”, RFC 1112, agosto 1989.

[RFC 1122] R. Braden, “Requirements for Internet Hosts—Communication Layers”, RFC 1122, octubre 1989.

[RFC 1123] R. Braden, ed., “Requirements for Internet Hosts—Application and Support”, *RFC-1123*, octubre 1989.

[RFC 1142] D. Oran, “OSI IS-IS Intra-domain Routing Protocol”, RFC 1142, febrero 1990.

[RFC 1190] C. Topolcic, “Experimental Internet Stream Protocol: Version 2 (ST-II),” RFC 1190, octubre 1990.

[RFC 1191] J. Mogul, S. Deering, “Path MTU Discovery”, RFC 1191, noviembre 1990.

[RFC 1213] K. McCloghrie, M. T. Rose, “Management Information Base for Network Management of TCP/IP-based internets: MIB-II”, RFC 1213, marzo 1991.

[RFC 1256] S. Deering, “ICMP Router Discovery Messages”, RFC 1256, septiembre 1991.

[RFC 1320] R. Rivest, “The MD4 Message-Digest Algorithm”, RFC 1320, abril 1992.

[RFC 1321] R. Rivest, “The MD5 Message-Digest Algorithm”, RFC 1321, abril 1992.

[RFC 1323] V. Jacobson, S. Braden, D. Borman, “TCP Extensions for High Performance”, RFC 1323, mayo 1992.

[RFC 1422] S. Kent, “Privacy Enhancement for Internet Electronic Mail: Part II: Certificate- Based Key Management”, RFC 1422.

[RFC 1547] D. Perkins, “Requirements for an Internet Standard Point-to-Point Protocol”, RFC 1547, diciembre 1993.

[RFC 1584] J. Moy, “Multicast Extensions to OSPF”, RFC 1584, marzo 1994.

[RFC 1633] R. Braden, D. Clark, S. Shenker, “Integrated Services in the Internet Architecture: an Overview”, RFC 1633, junio 1994.

[RFC 1636] R. Braden, D. Clark, S. Crocker, C. Huitema, “Report of IAB Workshop on Security in the Internet Architecture”, RFC 1636, noviembre 1994.

[RFC 1661] W. Simpson (ed.), “The Point-to-Point Protocol (PPP),” RFC 1661, julio 1994.

[RFC 1662] W. Simpson (ed.), “PPP in HDLC-like framing”, RFC 1662, julio 1994.

[RFC 1700] J. Reynolds y J. Postel, “Assigned Numbers”, RFC 1700, octubre 1994.

[RFC 1752] S. Bradner, A. Mankin, “The Recommendations for the IP Next Generation Protocol”, RFC 1752, enero 1995.

[RFC 1760] N. Haller, “The S/KEY One-Time Password System”, RFC 1760, febrero 1995.

[RFC 1918] Y. Rekhter, B. Moskowitz, D. Karrenberg, G. J. de Groot, E. Lear, “Address Allocation for Private Internets”, RFC 1918, febrero 1996.

[RFC 1930] J. Hawkinson, T. Bates, “Guidelines for Creation, Selection, and Registration of an Autonomous System (AS),” RFC 1930, marzo 1996.

[RFC 1938] N. Haller, C. Metz, “A One-Time Password System”, RFC 1938, mayo 1996.

[RFC 1939] J. Myers y M. Rose, “Post Office Protocol—Version 3”, RFC 1939, mayo 1996.

[RFC 1945] T. Berners-Lee, R. Fielding, H. Frystyk, “Hypertext Transfer Protocol—HTTP/1.0”, RFC 1945, mayo 1996.

[RFC 2003] C. Perkins, “IP Encapsulation within IP”, RFC 2003, octubre 1996.

[RFC 2004] C. Perkins, “Minimal Encapsulation within IP”, RFC 2004, octubre 1996.

[RFC 2018] M. Mathis, J. Mahdavi, S. Floyd, A. Romanow, “TCP Selective Acknowledgment Options”, RFC 2018, octubre 1996.

[RFC 2050] K. Hubbard, M. Kosters, D. Conrad, D. Karrenberg, J. Postel, “Internet Registry IP Allocation Guidelines”, RFC 2050, noviembre 1996.

[RFC 2104] H. Krawczyk, M. Bellare, R. Canetti, “HMAC: Keyed-Hashing for Message Authentication”, RFC 2104, febrero 1997.

[RFC 2131] R. Droms, “Dynamic Host Configuration Protocol”, RFC 2131, marzo 1997.

[RFC 2136] P. Vixie, S. Thomson, Y. Rekhter, J. Bound, “Dynamic Updates in the Domain Name System”, RFC 2136, abril 1997.

[RFC 2153] W. Simpson, “PPP Vendor Extensions”, RFC 2153, mayo 1997.

[RFC 2205] R. Braden, Ed., L. Zhang, S. Berson, S. Herzog, S. Jamin, “Resource ReSerVation Protocol (RSVP)—Version 1 Functional Specification”, RFC 2205, septiembre 1997.

[RFC 2210] J. Wroclawski, “The Use of RSVP with IETF Integrated Services”, RFC 2210, septiembre 1997.

[RFC 2211] J. Wroclawski, “Specification of the Controlled-Load Network Element Service”, RFC 2211, septiembre 1997.

[RFC 2215] S. Shenker, J. Wroclawski, “General Characterization Parameters for Integrated Service Network Elements”, RFC 2215, septiembre 1997.

[RFC 2246] T. Dierks y C. Allen, “The TLS Protocol”, RFC 2246, enero 1998.

[RFC 2253] M. Wahl, S. Kille, T. Howes, “Lightweight Directory Access Protocol (v3),” RFC 2253, diciembre 1997.

[RFC 2284] L. Blunk, J. Vollbrecht, “PPP Extensible Authentication Protocol (EAP),” RFC 2284, marzo 1998.

[RFC 2326] H. Schulzrinne, A. Rao, R. Lanphier, “Real Time Streaming Protocol (RTSP),” RFC 2326, abril 1998.

[RFC 2328] J. Moy, “OSPF Version 2”, RFC 2328, abril 1998.

[RFC 2420] H. Kummert, “The PPP Triple-DES Encryption Protocol (3DESE),” RFC 2420, septiembre 1998.

[RFC 2437] B. Kaliski, J. Staddon, “PKCS #1: RSA Cryptography Specifications, Version 2”, RFC 2437, octubre 1998.

[RFC 2453] G. Malkin, “RIP Version 2”, RFC 2453, noviembre 1998.

[RFC 2460] S. Deering, R. Hinden, “Internet Protocol, Version 6 (IPv6) Specification”, RFC 2460, diciembre 1998.

[RFC 2475] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, W. Weiss, “An Architecture for Differentiated Services”, RFC 2475, diciembre 1998.

[RFC 2578] K. McCloghrie, D. Perkins, J. Schoenwaelder, “Structure of Management Information Version 2 (SMIv2),” RFC 2578, abril 1999.

[RFC 2579] K. McCloghrie, D. Perkins, J. Schoenwaelder, “Textual Conventions for SMIv2”, RFC 2579, abril 1999.

[RFC 2580] K. McCloghrie, D. Perkins, J. Schoenwaelder, “Conformance Statements for SMIv2”, RFC 2580, abril 1999.

[RFC 2581] M. Allman, V. Paxson, W. Stevens, “ TCP Congestion Control”, RFC 2581, abril 1999.

[RFC 2597] J. Heinanen, F. Baker, W. Weiss, J. Wroclawski, “Assured Forwarding PHB Group”, RFC 2597, junio 1999.

[RFC 2616] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, T. Berners-Lee, R. Fielding, “Hypertext Transfer Protocol—HTTP/1.1”, RFC 2616, junio 1999.

[RFC 2663] P. Srisuresh, M. Holdrege, "IP Network Address Translator (NAT) Terminology and Considerations", RFC 2663.

[RFC 2702] D. Awduche, J. Malcolm, J. Agogbua, M. O'Dell, J. McManus, "Requirements for Traffic Engineering Over MPLS", septiembre 1999.

[RFC 2716] B. Aboba, D. Simon, "PPP EAP TLS Authentication Protocol", RFC 2716, octubre 1999.

[RFC 2733] J. Rosenberg, H. Schulzrinne, "An RTP Payload Format for Generic Forward Error Correction", RFC 2733, diciembre 1999.

[RFC 2827] P. Ferguson, D. Senie, "Network Ingress Filtering: Defeating Denial of Service Attacks which Employ IP Source Address Spoofing", RFC 2827, mayo 2000.

[RFC 2865] C. Rigney, S. Willens, A. Rubens, W. Simpson, "Remote Authentication Dial In User Service (RADIUS)," RFC 2865, junio 2000.

RFC 2960] R. Stewart, Q. Xie, K. Morneault, C. Sharp, H. Schwarzbauer, T. Taylor, I. Rytina, M. Kalla, L. Zhang, V. Paxson, "Stream Control Transmission Protocol", RFC 2960, octubre 2000.

[RFC 2961] L. Berger, D. Gan, G. Swallow, P. Pan, F. Tommasi, S. Molendini, "RSVP Refresh Overhead Reduction Extensions", RFC 2961, abril 2001.

[RFC 2988] V. Paxson, M. Allman, "Computing TCP's Retransmission Timer", RFC 2988, noviembre 2000.

[RFC 3007] B. Wellington, "Secure Domain Name System (DNS) Dynamic Update", RFC 3007, noviembre 2000.

[RFC 3022] P. Srisuresh, K. Egevang, "Traditional IP Network Address Translator (Traditional NAT)," RFC 3022, enero 2001.

[RFC 3022] P. Srisuresh, K. Egevang, "Traditional IP Network Address Translator (Traditional NAT)," RFC 3022, enero 2001.

[RFC 3031] E. Rosen, A. Viswanathan, R. Callon, "Multiprotocol Label Switching Architecture", RFC 3031, enero 2001.

[RFC 3032] E. Rosen, D. Tappan, G. Fedorkow, Y. Rekhter, D. Farinacci, T. Li, A. Conta, "MPLS Label Stack Encoding", RFC 3032, enero 2001.

[RFC 3052] M. Eder, S. Nag, "Service Management Architectures Issues and Review", RFC 3052, enero 2001.

[RFC 3139] L. Sanchez, K. McCloghrie, J. Saperia, "Requirements for Configuration Management of IP-Based Networks", RFC 3139, junio 2001.

[RFC 3168] K. Ramakrishnan, S. Floyd, D. Black, "The Addition of Explicit Congestion Notification (ECN) to IP", RFC 3168, septiembre 2001.

[RFC 3209] D. Awduche, L. Berger, D. Gan, T. Li, V. Srinivasan, G. Swallow, "RSVP-TE: Extensions to RSVP for LSP Tunnels", RFC 3209, diciembre 2001.

[RFC 3221] G. Huston, "Commentary on Inter-Domain Routing in the Internet", RFC 3221, diciembre 2001.

[RFC 3232] J. Reynolds, "Assigned Numbers: RFC 1700 is Replaced by an On-line Database", RFC 3232, enero 2002.

[RFC 3246] B. Davie, A. Charny, J.C.R. Bennet, K. Benson, J.Y. Le Boudec, W. Courtney, S. Davari, V. Firoiu, D. Stiliadis, "An Expedited Forwarding PHB (Per-Hop Behavior)," RFC 3246, marzo 2002.

[RFC 3260] D. Grossman, "New Terminology and Clarifications for Diffserv", RFC 3260, abril 2002.

[RFC 3261] J. Rosenberg, H. Schulzrinne, G. Carmarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley, E. Schooler, "SIP: Session Initiation Protocol", RFC 3261, julio 2002.

[RFC 3272] J. Boyle, V. Gill, A. Hannan, D. Cooper, D. Awduche, B. Christian, W.S. Lai, "Overview and Principles of Internet Traffic Engineering", RFC 3272, mayo 2002.

[RFC 3286] L. Ong, J. Yoakum, "An Introduction to the Stream Control Transmission Protocol (SCTP)," RFC 3286, mayo 2002.

[RFC 3344] C. Perkins, ed., "IP Mobility Support for IPv4", RFC 3344, octubre 2002.

[RFC 3346] J. Boyle, V. Gill, A. Hannan, D. Cooper, D. Awduche, B. Christian, W. S. Lai, "Applicability Statement for Traffic Engineering with MPLS", RFC 3346, agosto 2002.

[RFC 3376] B. Cain, S. Deering, I. Kouvelas, B. Fenner, A. Thyagarajan, "Internet Group Management Protocol, Version 3", RFC 3376, octubre 2002.

[RFC 3390] M. Allman, S. Floyd, C. Partridge, "Increasing TCP's Initial Window", RFC 3390, octubre 2002.

[RFC 3410] J. Case, R. Mundy, D. Partain, "Introduction and Applicability Statements for Internet Standard Management Framework", RFC 3410, diciembre 2002.

[RFC 3411] D. Harrington, R. Presuhn, B. Wijnen, "An Architecture for Describing Simple Network Management Protocol (SNMP) Management Frameworks", RFC 3411, diciembre 2002.

[RFC 3414] U. Blumenthal, "User-based Security Model (USM) for version 3 of the Simple Network Management Protocol (SNMPv3)," RFC 3414, diciembre 2002.

[RFC 3415] B. Wijnen, R. Presuhn, K. McCloghrie, "View-based Access Control Model (VACM) for the Simple Network Management Protocol (SNMP)," RFC 3415, diciembre 2002.

[RFC 3416] R. Presuhn, J. Case, K. McCloghrie, M. Rose, S. Waldbusser, "Version 2 of the Protocol Operations for the Simple Network Management Protocol (SNMP)," diciembre 2002.

[RFC 3417] R. Presuhn, "Transport Mappings for the Simple Network Management Protocol", (SNMP), RFC 3417, diciembre 2002.

[RFC 3439] R. Bush and D. Meyer, "Some internet architectural guidelines and philosophy", RFC 3439, diciembre 2003.

[RFC 3468] L. Andersson, G. Swallow, "The Multiprotocol Label Switching (MPLS) Working Group Decision on MPLS Signaling Protocols", RFC 3468, febrero 2003.

[RFC 3469] V. Sharma, Ed., F. Hellstrand, Ed, "Framework for Multi-Protocol Label Switching (MPLS)-based Recovery", RFC 3469, febrero 2003. ftp://ftp.rfc-editor.org/in-notes/rfc3469.txt

[RFC 3501] M. Crispin, "Internet Message Access Protocol—Version 4rev1", RFC 3501, marzo 2003.

[RFC 3550] H. Schulzrinne, S. Casner, R. Frederick, V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications", RFC 3550, julio 2003.

[RFC 3569] S. Bhattacharyya (ed.), "An Overview of Source-Specific Multicast (SSM)," RFC 3569, julio 2003.

[RFC 3588] P. Calhoun, J. Loughney, E. Guttman, G. Zorn, J. Arkko, "Diameter Base Protocol", RFC 3588, septiembre 2003.

[RFC 3618] B. Fenner, D. Meyer, Ed., "Multicast Source Discovery Protocol (MSDP)," RFC 3618, octubre 2003.

[RFC 3649] S. Floyd, "High Speed TCP for Large Congestion Windows", RFC 3649, diciembre 2003.

[RFC 3782] S. Floyd, T. Henderson, A. Gurtov, "The NewReno Modification to TCP's Fast Recovery Algorithm", RFC 3782, abril 2004.

[RFC 3973] A. Adams, J. Nicholas, W. Siadak, "Protocol Independent Multicast—Dense Mode (PIM-DM): Protocol Specification (Revised)," RFC 3973, enero 2005.

[RFC 4022] R. Raghunarayan, Ed., "Management Information Base for the Transmission Control Protocol (TCP)," RFC 4022, marzo 2005.

[RFC 4113] B. Fenner, J. Flick, "Management Information Base for the User Datagram Protocol (UDP)," RFC 4113, junio 2005.

[RFC 4213] E. Nordmark, R. Gilligan, "Basic Transition Mechanisms for IPv6 Hosts and Routers", RFC 4213, octubre 2005.

[RFC 4271] Y. Rekhter, T. Li, S. Hares, Ed., "A Border Gateway Protocol 4 (BGP-4)," RFC 4271, enero 2006.

[RFC 4272] S. Murphy, "BGP Security Vulnerabilities Analysis", RFC 4274, enero 2006.

[RFC4274] Meyer, D. y K. Patel, "BGP-4 Protocol Analysis", RFC 4274, enero 2006.

[RFC4276] Hares, S. y A. Retana, "BGP 4 Implementation Report", RFC 4276, enero 2006.

[RFC 4291] R. Hinden, S. Deering, "IP Version 6 Addressing Architecture", RFC 4291, febrero 2006.

[RFC 4293] S. Routhier, Ed. "Management Information Base for the Internet Protocol (IP)," RFC 4293, abril 2006.

[RFC 4301] S. Kent, K. Seo, "Security Architecture for the Internet Protocol", RFC 4301, diciembre 2005.

[RFC 4302] S. Kent, "IP Authentication Header", RFC 4302, diciembre 2005.

[RFC 4303] S. Kent, "IP Encapsulating Security Payload (ESP)," RFC 4303, diciembre 2005.

[RFC 4305] D. Eastlake, "Cryptographic Algorithm Implementation Requirements for Encapsulating Security Payload (ESP) and Authentication Header (AH)," RFC 4305, diciembre 2005.

[RFC 4340] E. Kohler, M. Handley, S. Floyd, "Datagram Congestion Control Protocol (DCCP)," RFC 4340, marzo 2006.

[RFC 4443] A. Conta, S. Deering, M. Gupta, Ed., "Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification", RFC 4443, marzo 2006.

[RFC 4346] T. Dierks, E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.1", RFC 4346, abril 2006.

[RFC 4502] S. Waldbusser, "Remote Network Monitoring Management Information Base Version 2", RFC 4502, mayo 2006.

[RFC 4601] B. Fenner, M. Handley, H. Holbrook, I. Kouvelas, "Protocol Independent Multicast—Sparse Mode (PIM-SM): Protocol Specification (Revised)," RFC 4601, agosto 2006.

[RFC 4607] H. Holbrook, B. Cain, "Source-Specific Multicast for IP", RFC 4607, agosto 2006.

[RFC 4611] M. McBride, J. Meylor, D. Meyer, "Multicast Source Discovery Protocol (MSDP) Deployment Scenarios", RFC 4611, agosto 2006.

[RFC 4632] V. Fuller, T. Li, "Classless Inter-domain Routing (CIDR): The Internet Address Assignment and Aggregation Plan", RFC 4632, agosto 2006.

[RFC 5000] RFC editor, "Internet Official Protocol Standards", RFC 5000, mayo 2008.

[RFC 5110] P. Savola, "Overview of the Internet Multicast Routing Architecture", RFC 5110, enero 2008.

[RFC 5218] D. Thaler, B. Aboba, "What Makes For a Successful Protocol?," RFC 5218, julio 2008.

[RFC 5321] J. Klensin, "Simple Mail Transfer Protocol", RFC 5321, octubre 2008.

[RFC 5322] P. Resnick, Ed., "Internet Message Format", RFC 5322 octubre 2008.

[RFC 5348] S. Floyd, M. Handley, J. Padhye, J.Widmer, "TCP Friendly Rate Control (TFRC): Protocol Specification", RFC 5348, septiembre. 2008.

[RFC 5411] J Rosenberg, "A Hitchhiker's Guide to the Session Initiation Protocol (SIP)," RFC 5411, febrero 2009.

[Rhee 1998] I. Rhee, "Error Control Techniques for Interactive Low-bit Rate Video Transmission over the Internet", *Proc. 1998 ACM SIGCOMM* (Vancouver BC, agosto 1998).

[Roberts 1967] L. Roberts, T. Merril, "Toward a Cooperative Network of Time-Shared Computers", *AFIPS Fall Conference,* octubre 1966.

[Roberts 2004] J. Roberts, "Internet Traffic, QoS and Pricing", Proceedings of the IEEE, Vol. 92, Nº 9 (septiembre 2004), págs. 1389–1399.

[Rom 1990] R. Rom, M. Sidi, *Multiple Access Protocols: Performance and Analysis,* Springer-Verlag, Nueva York, 1990.

[Root-servers 2009] http://www.root-servers.org/

[Rose 1996] M. Rose, *The Simple Book: An Introduction to Internet Management, Revised Second Edition,* Prentice Hall, Englewood Cliffs, NJ, 1996.

[Rosenberg 2000] J. Rosenberg, L. Qiu, H. Schulzrinne, "Integrating Packet FEC into Adaptive Playout Buffer Algorithms on the Internet", *Proc. 2000 IEEE Infocom* (Tel Aviv, Israel, abril 2000).

[Ross 1995] K. W. Ross, *Multiservice Loss Models for Broadband Telecommunication Networks,* Springer, Berlín, 1995.

[Ross 2009] K. W. Ross, PowerPoint slides on network Security, http://cis.poly.edu/~ross

[Rowston 2001] A. Rowston, P. Druschel, "Pastry: Scalable, Distributed Object Location and Routing for Large-Scale Peer-to-Peer Systems", in *Proc. 2001 IFIP/ACM Middleware*, Heidelberg, Alemania, 2001.

[RSA 1978] R. Rivest, A. Shamir, L. Adelman, "A Method for Obtaining Digital Signatures and Public-key Cryptosystems", *Communications of the ACM,* Vol. 21, Nº 2, págs. 120–126, febrero 1978.

[RSA Fast 2009] RSA Laboratories, "How fast is RSA?" http://www.rsa.com/rsalabs/node .asp?id=2215

[RSA Key 2009] RSA Laboratories, "How large a key should be used in the RSA Crypto system?" http://www.rsa.com/rsalabs/node.asp?id=2218

[Rubenstein 1998] D. Rubenstein, J. Kurose, D. Towsley, "Real-Time Reliable Multicast Using Proactive Forward Error Correction", *Proceedings of NOSSDAV '98* (Cambridge, UK, julio 1998).

[Rubin 2001] A. Rubin, *White-Hat Security Arsenal: Tackling the Threats,* Addison-Wesley, 2001.

[Ruiz-Sanchez 2001] M. Ruiz-Sánchez, E. Biersack, W. Dabbous, "Survey and Taxonomy of IP Address Lookup Algorithms", *IEEE Network Magazine,* Vol. 15, Nº 2, págs. 8–23, marzo/abril 2001.

[Saltzer 1984] J. Saltzer, D. Reed, D. Clark, "End-to-End Arguments in System Design", *ACM Transactions on Computer Systems (TOCS),* Vol. 2, Nº 4 (noviembre 1984).

[Saroiu 2002] S. Saroiu, P.K. Gummadi, S.D. Gribble, "A Measurement Study of Peer-to-Peer File Sharing Systems", *Proc. of Multimedia Computing and Networking (MMCN),* 2002.

[Saroiu 2002b] S. Saroiu, K. P. Gummadi, R. J. Dunn, S. D. Gribble y H. M. Levy, "An Analysis of Internet Content Delivery Systems", *USENIX OSDI,* 2002.

[Saydam 1996] T. Saydam, T. Magedanz, "From Networks and Network Management into Service and Service Management", *Journal of Networks and System Management,* Vol. 4, Nº 4 (diciembre 1996), págs. 345–348.

[Schiller 2003] J. Schiller, *Mobile Communications* 2ª edición, Addison Wesley, 2003.

[Schneier 1995] B. Schneier, *Applied Cryptography: Protocols, Algorithms, and Source Code in C,* John Wiley and Sons, 1995.

[Schulzrinne 1997] H. Schulzrinne, "A Comprehensive Multimedia Control Architecture for the Internet", *NOSSDAV'97 (Network and Operating System Support for Digital Audio and Video)* (St. Louis, MO, mayo 1997).

[Schulzrinne-RTP 2009] Sitio RTP de Henning Schulzrinne: http://www.cs.columbia.edu/~hgs/rtp

[Schulzrinne-RTSP 2009] Sitio de RTSP de Henning Schulzrinne: http://www.cs.columbia.edu/~hgs/rtsp

[Schulzrinne-SIP 2009] Sitio SIP de Henning Schulzrinne: http://www.cs.columbia.edu/~hgs/sip

[Schwartz 1977] M. Schwartz, *Computer-Communication Network Design and Analysis*, Prentice-Hall, Englewood Cliffs, N.J., 1997.

[Schwartz 1980] M. Schwartz, *Information, Transmission, Modulation, and Noise,* McGraw Hill, NY, NY 1980.

[Schwartz 1982] M. Schwartz, "Performance Analysis of the SNA Virtual Route Pacing Control", *IEEE Transactions on Communications,* Vol. 30, Nº 1 (enero 1982), págs. 172–184.

[Scourias 2009] J. Scourias, “Overview of the Global System for Mobile Communications: GSM.” http://www.privateline.com/PCS/GSM0.html

[Segaller 1998] S. Segaller, *Nerds 2.0.1, A Brief History of the Internet,* TV Books, Nueva York, 1998.

[Shacham 1990] N. Shacham, P. McKenney, “Packet Recovery in High-Speed Networks Using Coding and Buffer Management”, *Proc. 1990 IEEE Infocom* (San Francisco, CA, abril 1990), págs. 124–131.

[Sharma 2003] P. Sharma, E, Perry, R. Malpani, “IP Multicast Operational Network management: Design, Challenges, and Experiences”, *IEEE Network Magazine,* marzo 2003, págs. 49–55.

[Sidor 1998] D. Sidor, “TMN Standards: Satisfying Today’s Needs While Preparing for Tomorrow”, *IEEE Communications Magazine,* Vol. 36, Nº 3 (marzo 1998), págs. 54–64.

[SIP Software 2009] H. Schulzrinne Software Package site, http://www.cs.columbia.edu/IRT/software

[Skoudis 2004] E. Skoudis, L. Zeltser, *Malware: Fighting Malicious Code*, Prentice Hall, 2004.

[Skoudis 2006] E. Skoudis, T. Liston, *Counter Hack Reloaded: A Step-by-Step Guide to Computer Attacks and Effective Defenses (2ª ed.),* Prentice Hall, 2006.

[Skype 2009] Página principal de Skype: www.skype.com

[SMIL 2009] Página principal de W3C Synchronized Multimedia: http://www.w3.org/AudioVideo

[Snort 2009] Página principal de Sourcefire Inc., Snort: http://http://www.snort.org/

[Solari 1997] S. J. Solari, *Digital Video and Audio Compression,* McGraw Hill, NY, NY, 1997.

[Solensky 1996] F. Solensky, “IPv4 Address Lifetime Expectations”, in *IPng: Internet Protocol Next Generation* (S. Bradner, A. Mankin, ed.), Addison-Wesley, Reading, MA, 1996.

[Spragins 1991] J. D. Spragins, *Telecommunications Protocols and Design,* Addison-Wesley, Reading, MA, 1991.

[Sprint 2009] Sprint Corp., “Dedicated Internet Access Service Level Agreements”, http:// www.sprint.com/business/resources/dedicated_internet_access.pdf

[Srikant 2004] R. Srikant, *The Mathematics of Internet Congestion Control,* Birkhauser, 2004

[Srinivasan 1999] V. Srinivasan y G. Varghese, “Fast Address Lookup Using Controlled Prefix Expansion”, *ACM Transactions Computer Sys.*, Vol. 17, Nº 1 (febrero 1999), págs. 1–40.

[Sripanidkulchai 2004] K. Sripanidkulchai, B. Maggs y H. Zhang, “An analysis of live streaming workloads on the Internet”, *Proc. 4th ACM SIGCOMM Internet Measurement Conference* (Taormina, Sicilia, Italia), págs. 41–54, 2004.

[Stallings 1993] W. Stallings, *SNMP, SNMP v2, and CMIP The Practical Guide to Network Management Standards,* Addison-Wesley, Reading, MA, 1993.

[Stallings 1999] W. Stallings, *SNMP, SNMPv2, SNMPv3, and RMON 1 and 2,* Addison-Wesley, Reading, MA, 1999.

[Steinder 2002] M. Steinder, A. Sethi, “Increasing robustness of fault localization through analysis of lost, spurious, and positive symptoms”, *Proc. 2002 IEEE Infocom.*

[Stevens 1990] W. R. Stevens, *Unix Network Programming,* Prentice-Hall, Englewood Cliffs, NJ.

[Stevens 1994] W. R. Stevens, *TCP/IP Illustrated, Vol. 1: The Protocols,* Addison-Wesley, Reading, MA, 1994.

[Stevens 1997] W.R. Stevens, *Unix Network Programming, Volume 1: Networking APIs-Sockets and XTI,* 2nd edition, Prentice-Hall, Englewood Cliffs, NJ, 1997.

[Stewart 1999] J. Stewart, *BGP4: Interdomain Routing in the Internet,* Addison-Wesley, 1999.

[Stoica 2001] I. Stoica, R. Morris, D. Karger, M.F. Kaashoek, H. Balakrishnan, “Chord: A Scalable Peer-to-Peer Lookup Service for Internet Applications”, *Proc. 2001 ACM SIGCOMM* (San Diego, CA, agosto 2001).

[Stone 1998] J. Stone, M. Greenwald, C. Partridge, J. Hughes, "Performance of Checksums and CRC's Over Real Data", *IEEE/ACM Transactions on Networking,* Vol. 6, Nº 5 (octubre 1998), págs. 529–543.

[Stone 2000] J. Stone, C. Partridge, "When Reality and the Checksum Disagree", *Proc. 2000 ACM SIGCOMM* (Estocolmo, Suecia, agosto 2000).

[Strayer 1992] W. T. Strayer, B. Dempsey, A. Weaver, *XTP: The Xpress Transfer Protocol,* Addison-Wesley, Reading, MA, 1992.

[Stubblefield 2002] A. Stubblefield, J. Ioannidis, A. Rubin, "Using the Fluhrer, Mantin, and Shamir Attack to Break WEP", *Proceedings of 2002 Network and Distributed Systems Security Symposium* (2002), 17–22.

[Subramanian 2000] M. Subramanian, *Network Management: Principles and Practice,* Addison-Wesley, Reading, MA, 2000.

[Subramanian 2002] L. Subramanian, S. Agarwal, J. Rexford, R. Katz, "Characterizing the Internet Hierarchy from Multiple Vantage Points", *Proc. 2002 IEEE Infocom.*

[Sundaresan 2006] K.Sundaresan, K. Papagiannaki, "The Need for Cross-layer Information in Access Point Selection", *Proc. 2006 ACM Internet Measurement Conference* (Río de Janeiro, octubre 2006).

[Su 2006] A.-J. Su, D. Choffnes, A. Kuzmanovic y F. Bustamante, "Drafting Behind Akamai" *ACM SIGCOMM,* septiembre 2006.

[Suh 2006] K. Suh, D. R. Figueiredo, J. Kurose y D. Towsley, "Characterizing and detecting relayed traffic: A case study using Skype", *Proc. 2006 IEEE Infocom* (Barcelona, España, abril 2006).

[Sunshine 1978] C. Sunshine, Y. Dalal, "Connection Management in Transport Protocols", *Computer Networks,* North-Holland, Amsterdam, 1978.

[TechnOnLine 2009] TechOnLine, "Protected Wireless Networks", online webcast tutorial, http:/ /www.techonline.com/community/tech_topic/internet/21752

[Thaler 1997] D. Thaler and C. Ravishankar, "Distributed Center-Location Algorithms", *IEEE Journal on Selected Areas in Communications,* Vol. 15, Nº 3, (abril 1997), págs. 291–303.

[Think 2009] Technical History of Network Protocols, "Cyclades", http://www.cs.utexas .edu/users/ chris/think/Cyclades/index.shtml

[Thottan 1998] M. Thottan, C. Ji, "Proactive Anomaly Detection Using Distributed Intelligent Agents", *IEEE Network Magazine,* Vol. 12, Nº 5 (sept./oct. 1998), págs. 21–28.

[Tobagi 1990] F. Tobagi, "Fast Packet Switch Architectures for Broadband Integrated Networks", *Proc. of the IEEE,* Vol. 78, Nº 1 (enero 1990), págs. 133–167.

[TOR 2009] Tor: Anonymity Online, http://www.torproject.org

[Turner 1988] J. S. Turner "Design of a Broadcast packet switching network", *IEEE Transactions on Communications,* Vol. 36, Nº 6 (junio 1988), págs. 734–743.

[Turner 2009] B. Turner, "2G, 3G, 4G Wireless Tutorial", http://blogs.nmscommunications .com/ communications/2008/10/2g-3g-4g-wireless-tutorial.html

[UPnP Forum 2009] Página principal de UPnP Forum: http://www.upnp.org/

[van der Berg 2008] R. van der Berg, "How the 'Net works: an introduction to peering and transit", http://arstechnica.com/guides/other/peering-and-transit.ars

[Varghese 1997] G. Varghese, A. Lauck, "Hashed and Hierarchical Timing Wheels: Efficient Data Structures for Implementing a Timer Facility", *IEEE/ACM Transactions on Networking,* Vol. 5, Nº 6 (diciembre 1997), págs. 824–834.

[Vasudevan 2006] S. Vasudevan, C. Diot, J. Kurose, D. Towsley, "Facilitating Access Point Selection in IEEE 802.11 Wireless Networks", *Proc. 2005 ACM Internet Measurement Conference* (San Francisco CA, octubre 2005).

[Verizon 2009] Verizon, “US Products and Services”, http://www.verizonbusiness.com/terms/us/products/

[Verizon FIOS 2009] FIOS FAQ, http://www22.verizon.com/Residential/FiOSInternet/FAQ/FAQ.htm

[Verma 2001] D. C. Verma, *Content Distribution Networks: An Engineering Approach,* John Wiley, 2001.

[Villamizar 1994] C. Villamizar, C. Song. “High performance tcp in ansnet”, *ACM SIGCOMM Computer Communications Review*, Vol. 24, Nº 5 (1994), págs. 45–60.

[Viterbi 1995] A. Viterbi, *CDMA: Principles of Spread Spectrum Communication,* Addison-Wesley, Reading, MA, 1995.

[Voydock 1983] V. L. Voydock, S.T. Kent, “Security Mechanisms in High-Level Network Protocols”, *ACM Computing Surveys,* Vol. 15, Nº 2 (junio 1983), págs. 135–171.

[W3C 1995] The World Wide Web Consortium, “A Little History of the World Wide Web”, 1995. http://www.w3.org/History.html

[Wakeman 1992] I. Wakeman, J. Crowcroft, Z. Wang, D. Sirovica, “Layering Considered Harmful”, *IEEE Network,* enero 1992, págs. 20–24.

[Waldvogel 1997] M. Waldvogel et al., “Scalable High Speed IP Routing Lookup”, *Proc. 1997 ACM SIGCOMM* (Cannes, Francia, septiembre 1997).

[Walker 2000] J. Walker, “IEEE P802.11 Wireless LANs, Unsafe at Any Key Size; An Analysis of the WEP Encapsulation”, Oct. 2000, http://www.drizzle.com/~aboba/IEEE/0-362.zip

[Wall 1980] D. Wall, *Mechanisms for Broadcast and Selective Broadcast,* Ph.D. thesis, Stanford University, junio 1980.

[Wang 2004] B. Wang, J. Kurose, P. Shenoy, D. Towsley, “Multimedia Streaming via TCP: An Analytic Performance Study”, *Proc. ACM Multimedia Conf.* (NY, NY, octubre 2004).

[Weatherspoon 2000] S. Weatherspoon, “Overview of IEEE 802.11b Security”, *Intel Technology Journal,* (2nd Quarter 2000), http://download.intel.com/technology/itj/q22000/pdf/art_5.pdf

[Wei 2005] W. Wei, B. Wang, C. Zhang, J. Kurose, D. Towsley, “Classification of Access Network Types: Ethernet, Wireless LAN, ADSL, Cable Modem or Dialup?,” *Proc. 2005 IEEE Infocom* (abril 2005).

[Wei 2006] W. Wei, C. Zhang, H. Zang, J. Kurose, D. Towsley, “Inference and Evaluation of Split-Connection Approaches in Cellular Data Networks”, *Proc. Active and Passive Measurement Workshop* (Adelaide, Australia, marzo 2006).

[Wei 2007] D. X. Wei, C. Jin, S. H. Low, S. Hegde, “FAST TCP: Motivation, Architecture, Algorithms, Performance”, *IEEE/ACM Transactions on Networking,* 2007.

[Weiser 1991] M. Weiser, “The Computer for the Twenty-First Century”, *Scientific American* (septiembre 1991): 94–10. http://www.ubiq.com/hypertext/weiser/SciAmDraft3.html

[Wigle.net 2009] Wireless Geographic Logging Engine, http://www.wigle.net

[Williams 1993] R. Williams, “A Painless Guide to CRC Error Detection Algorithms”, http://www.ross.net/crc/crcpaper.html

[WiMax Forum 2009] WiMax Forum, http://www.wimaxforum.org

[Wireshark 2009] Página principal de Wireshark: http://www.wireshark.org

[Wischik 2005] D. Wischik, N. McKeown, “Part I: Buffer Sizes for Core Routers”, *ACM SIGCOMM Computer Communications Review*, Vol. 35, Nº 3, julio 2005.

[Woo 1994] T. Woo, R. Bindignavle, S. Su, S. Lam, “SNP: an interface for secure network programming”, *Proc. 1994 Summer USENIX,* Boston, MA, junio 1994, págs. 45–58.

[Wood 2009] L. Wood, “Lloyds Satellites Constellations”, http://www.ee.surrey.ac.uk/Personal/L.Wood/constellations/iridium.html

[Xanadu 2009] Página principal del proyecto Xanadu: http://www.xanadu.com/

[Xiao 2000] X. Xiao, A. Hannan, B. Bailey, L. Ni, "Traffic Engineering with MPLS in the Internet", *IEEE Network*, marzo/abril 2000.

[Xie 2008] H. Xie, Y.R. Yang, A. Krishnamurthy, Y. Liu, A. Silberschatz, "P4P: Provider Portal for Applications", *Proc. ACM SIGCOMM* (Seattle, agosto 2008).

[Yannuzzi 2005] M. Yannuzzi, X. Masip-Bruin, O. Bonaventure, "Open Issues in Interdomain Routing: A Survey", *IEEE Network Magazine,* nov./dic. 2005.

[Yavatkar 1994] R. Yavatkar, N. Bhagwat, "Improving End-to-End Performance of TCP over Mobile Internetworks", *Proc. Mobile 94 Workshop on Mobile Computing Systems and Applications,* diciembre 1994.

[Youtube 2009] Página principal de Youtube: www.youtube.com

[Yu 2006] H. Yu, M. Kaminsky, P. B. Gibbons y A. Flaxman, SybilGuard: Defending Against Sybil Attacks via Social Networks, *Proc. 2006 ACM* (Pisa, Italia, septiembre 2006).

[Zegura 1997] E. Zegura, K. Calvert, M. Donahoo, "A Quantitative Comparison of Graph-based Models for Internet Topology", *IEEE/ACM Transactions on Networking,* Vol. 5, Nº 6, (diciembre 1997). Véase también http://www.cc.gatech.edu/projects/gtim para obtener un paquete software que permite generar redes con una estructura de red terminal de tránsito.

[Zhang 1993] L. Zhang, S. Deering, D. Estrin, S. Shenker, D. Zappala, "RSVP: A New Resource Reservation Protocol", *IEEE Network Magazine,* Vol. 7, Nº 9 (septiembre 1993), págs. 8–18.

[Zhao 2004] B. Y. Zhao, L. Huang, J. Stribling, S. C. Rhea, A. D. Joseph, J. Kubiatowicz, "Tapestry: A Resilient Global-scale Overlay for Service Deployment", *IEEE Journal on Selected Areas in Communications,* Vol. 22, Nº 1 (enero 2004).

[Zimmerman 1980] H. Zimmerman, "OS1 Reference Model-The ISO Model of Architecture for Open Systems Interconnection", *IEEE Transactions on Communications,* Vol. 28, Nº 4 (abril 1980), págs. 425–432.

[Zimmermann 2009] P. Zimmermann, "Why do you need PGP?" http://www.pgpi.org/doc/whypgp/en/

Índice

G

H

I

Q

R



Redes de computadoras es el principal libro de texto en el aprendizaje de los aspectos básicos de redes. Los profesores Jim Kurose y Keith Ross hacen una presentación muy atractiva de todos los temas, utilizando un enfoque desde arriba hacia abajo en el tratamiento de redes e Internet.

Esta edición conserva el énfasis original sobre los paradigmas de capa de aplicación, la programación de aplicaciones y los protocolos de capa superior, fomentando la transmisión de la experiencia con conceptos sobre protocolos y redes.

Este libro incluye una suscripción de seis meses al sitio web que complementa al libro. Esta suscripción gratuita proporciona complementos de apoyo al estudiante, incluyendo:

- Once prácticas de laboratorio con Wireshark.
- Nueve prácticas de programación.
- Applets en Java que ilustran diversos conceptos clave de las redes.
- Cuestionarios interactivos que le ayudarán a evaluar su grado de comprensión de los temas estudiados.
- Interesantes enlaces a otros recursos.

Addison Wesley
es un sello editorial de



www.pearsoneducacion.com

